

Enterprise Integration

Kick off & *Introduction*

Prof. Sérgio Guerreiro

Sergio.guerreiro@tecnico.ulisboa.pt

Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

Faculty

Faculty



Sérgio Guerreiro Senior Lecturer
sergio.guerreiro@tecnico.ulisboa.pt



Flávio Martins
flavio.f.martins@tecnico.ulisboa.pt

Major IE 2025 Goals

- To learn what enterprise integration is
- To understand that integration and cloud are fundamental parts of the digital economy
- To have a practical experience with the “cloud” integration technologies

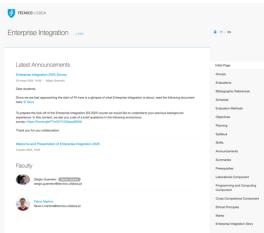
In the end, to understand the information systems of digital (big) companies

Detailed Course Program

The course aims at providing a coherent structure of integration topics that can be found in different parts of the ACM/AIS IS 2010 curriculum, such as “Enterprise Systems” and “Application Development”

- Evolution of information systems. Functions. Evolution. Service.
- Introduction to integration platforms. Messages. Transformations. Schemas. Ports. Adapter. Orchestration. Business rule.
- Messaging systems. Transactions. Correlations. Platforms.
- Message brokers. Publish-subscribe. Streaming. Asynchronous.
- Adapters. 3-tier. Methods. Database. Files. User interface. Dynamic invocations. Web services.
- Micro-services, services and SOA. Applications. Composition. Catalog. Business processes. Service design. Human workflows.
- Service orchestrations. Flow. Decisions. Loops. Events. Exceptions. Transactions.
- Inter-organizational integration. Electronic data exchange. Supply chain management. E-commerce. Negotiation. Internet of things.

Hyperautomation is a transversal concern in all those aspects. How to do it faster and better?



Study artifacts

- Academic page for course @ Fénix:
<https://fenix.tecnico.ulisboa.pt/disciplinas/SEI36/2024-2025/2-semestre>
- Q&A @ Moodle: <https://moodle.dei.tecnico.ulisboa.pt/course/view.php?id=6273>
- AWS Academy classroom for each sprint -> invitation sent for sprint 1 (using your fenix registered email)
 - SPRINT 1 <https://awsacademy.instructure.com/courses/117013>
 - SPRINT 2 <https://awsacademy.instructure.com/courses/117014>
- Slides + Project + Scenarios + tutorials are already available @ Fénix



Class dynamics

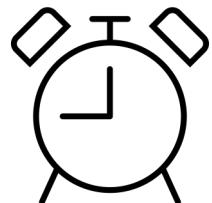
This is a **hands-on** course using **flipped learning**:

1. Understand the IE concepts **at concept classes**
2. Execute the tutorial, **at home**
3. Watch the movie tutorial, if needed, **at home**
4. Deploy the scenarios and adapt them, **at home**
5. Bring questions/problems to apply the tutorial to the **LAB** and **project** classes



Weekly schedule

	Seg 5/5	Ter 5/6	Qua 5/7	Qui 5/8	Sex 5/9
07:00					
08:00					
09:00		08:30 - 10:00 L F8	08:30 - 10:30 T EA3	08:30 - 10:00 L V1.25	
10:00	09:30 - 11:30 T EA3	10:00 - 11:30 L F8		10:00 - 11:30 L V1.12	
11:00					



The slots are organized as **concept classes** and **lab classes** and **project classes**

Course Planning

			Content	Tutorial Guidelines	Tutorial movies	Project Deliverables	
28/04/2025	Week 1	EA3	Concepts 00 - Intro / 00 - Enterprise Integration				
29/04/2025		FB + F8	LAB Kafka in the cloud	P1-Kafka-in-AWSAcademy.pdf	EC2-VM Creation and Accessing Kafka Installation EC2 Using Kafka EC2		
30/04/2025		EA3	Concepts 01 - MoM - Kafka - Streams				
01/05/2025		V1.25 + V1.12	<i>Holiday: Labor Day</i>				
05/05/2025	Week 2	EA3	LAB Kafka in the cloud (cont.) Terraform	P1-Kafka-in-AWSAcademy.pdf P2-HyperAutomation-TERRAFORM.pdf	EC2-VM Creation and Accessing Kafka Installation EC2 Using Kafka EC2		
06/05/2025		FB + F8	LAB Kafka clustering Kafka Streams	P4-Kafka-Streams-AWS.pdf P3-Kafka-Distributed.pdf	KafkaCluster-Configuration		
07/05/2025		EA3	Concepts 02 - SOA & Microservices & Reactive & Quarkus				
08/05/2025		V1.25 + V1.25	LAB Quarkus	P6-Quarkus.pdf			
12/05/2025	Week 3	EA3	LAB Quarkus with Docker (introductory)	P6-Quarkus.pdf			
13/05/2025		FB + F8	LAB Quarkus: REST & DB & Swagger	P6-Quarkus.pdf P5-RDS-database.pdf			
14/05/2025		EA3	Evaluation Revision + MAP45				
15/05/2025		V1.25 + V1.12	Project Sprint 1 support		Saturday 17/5/2025 12h Sprint 1 (15%)		
19/05/2025	Week 4	EA3	Concepts 03 - Containers & Cloud				
20/05/2025		FB + F8	LAB AWS LAMBDA	P8-Lambda-AWS.pdf	Develop Deploy and Invoke a Lambda microservice		
21/05/2025		EA3	Concepts 04 - Processes				
22/05/2025		V1.25 + V1.12	LAB Sprint 1 Patterns feedback				
26/05/2025	Week 5	EA3	LAB Camunda platform	P7-BPMN-CAMUNDA.pdf Scenario 1	Camunda-Overview A full example of a executable BPMN collaboration in Camunda engine		
27/05/2025		FB + F8	LAB Business process design & implementation Business process Integration & DMN	P7-BPMN-CAMUNDA.pdf Scenario 1	Camunda-Overview A full example of a executable BPMN collaboration in Camunda engine		
28/05/2025		EA3	Concepts 05 - API management and Gateway systems				
29/05/2025		V1.25 + V1.12	LAB Authentication & credentials APIs Deployment & management	P9-KONG.pdf Scenario2	Kong APIGateway Configuration and Invocation		
02/06/2025	Week 6	EA3	Concepts 05 - Identity management & B2B & Other aspects				
03/06/2025		FB + F8	LAB Authentication & credentials / APIs Deployment & management (cont.) Ollama deployment in EC2	P9-KONG.pdf Scenario2 P2-HyperAutomation-TERRAFORM.pdf	Kong APIGateway Configuration and Invocation		
04/06/2025		EA3	Evaluation Revision + MAP45				
05/06/2025		V1.25 + V1.25	Project Sprint 2 Support	Scenario 3			
09/06/2025	Week 7	EA3	Project Sprint 2 Support				
10/06/2025		FB + F8	<i>Holiday: Day of Portugal, Camões and Portuguese Communities</i>				
11/06/2025		EA3	Project Sprint 2 support				
12/06/2025		V1.25 + V1.25	Project Sprint 2 Support		Saturday 14/6/2025 12h Sprint 2 (25%)		
	Week 8	project discussion					
		Normal Season Exam - 26/6/2025 10:30					
		Second Season Exam - 10/7/2025 15:30					
		Special Season Exam - 23/7/2025 18:00					

	amount of classes
LAB	12
Concepts	7
project	5
evaluation	2

Links for tutorial movies

(use IST ID Gsuite only for accessing)

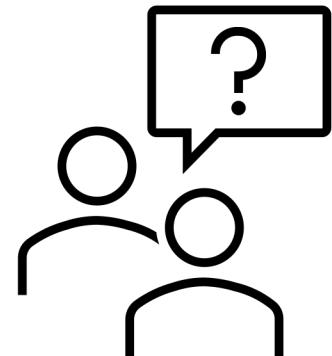
How to provision an EC2 VM?	https://drive.google.com/file/d/1GicH9SufhMnl2oCSRGjYJteNRxPFKtH-/view?usp=sharing
How to install Kafka in an EC2 VM?	https://drive.google.com/file/d/1zp_tEYVgq1o9psL7y064qMDWNoQmKuqd/view?usp=sharing
How to use Kafka in an EC2 VM?	https://drive.google.com/file/d/1h5qW2X9AM7aEVzuMv7FhfTcXgyC7gCU/view?usp=sharing
How to enforce a Kafka Cluster?	https://drive.google.com/file/d/1da2aEJ4HywPNgCQCH4tztLRRR9wqK2t3/view?usp=sharing
How to develop, deploy, and execute an AWS Lambda service?	https://drive.google.com/file/d/1rAr27ZAZRdlw9sVCmThUNa_6zietEV5F/view?usp=sharing
What is Camunda?	https://drive.google.com/file/d/1b-j7-OaS1gWM1mJaQGbjt6vWU9L2MYme/view?usp=sharing
How to model, develop, deploy, and execute a business process using Camunda engine?	https://drive.google.com/file/d/1Yxqch1nGm7hOD3Cpz1eiiQeuzmK5Tt2h/view?usp=sharing
How to install and operate Kong in an EC2 VM?	https://drive.google.com/file/d/1vryBWG4BqgLt4cL1uUXwXC_f3Emma-vS/view?usp=sharing

Questions classes

By Zoom: <https://videoconf-colibri.zoom.us/j/2785489605?pwd=SXFmNzdMZXAA2TTIiZ09QSFZPUXJhUT09>

Send an email requesting it beforehand.

Tuesday and Thursday at 16:30



Evaluation

The final grade is given by:

NotaFinal época Normal = round(MAX(0,2 * NExameNormal + 0,40 * NMAP45 ; 0,60 * ExameNormal) + 0,40 * NProjects)

OR

NotaFinal época Recurso = round(0,60* NExameRecurso + 0,40*NProjects)

OR

NotaFinal época Especial = round(0,60* NExameEspecial + 0,40*NProjects)

Where:

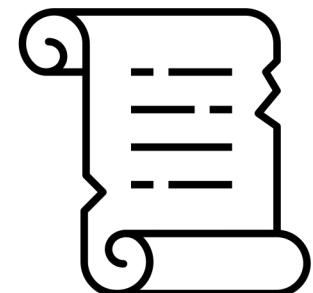
2 * NMAP45 are done during classes

NExameNormal, NMAP45, NExameRecurso and NExameEspecial requires $\geq 8,0$ values

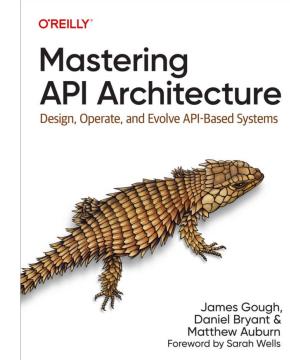
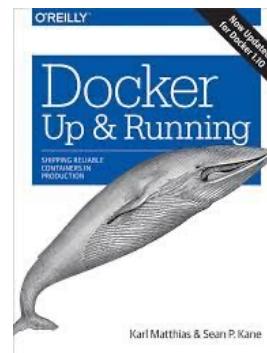
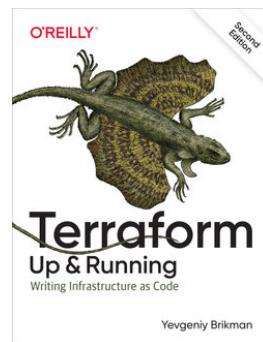
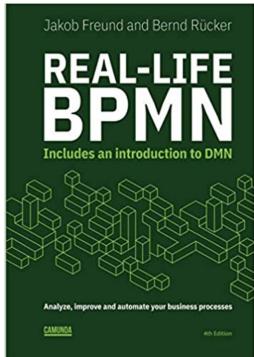
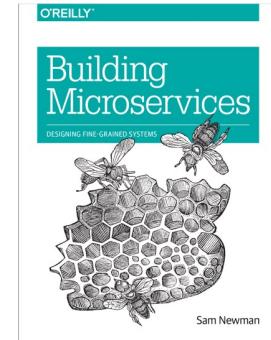
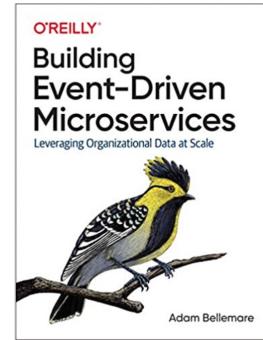
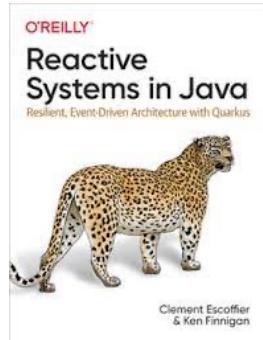
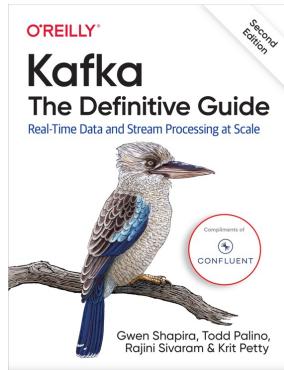
NProjects consists of 2 sprints (15% - 25% for each sprint)

NProjects $\geq 8,0$ values -> Project is done by groups of 2 students

NotaFinal época Normal, NotaFinal época Recurso, NotaFinal época Especial $\geq 9,5$ values



Books



Books

Gwen Shapira, Todd Palino, Rajini Sivaram & Krit Petty (2021) Kafka: The Definitive Guide, O'Reilly Media, Inc., ISBN: 978-1-491-99065-0

Escoffier, C., Finnigan, K. (2021). Reactive Systems in Java, O'Reilly Media, Inc., ISBN: 9781492091721

Freund J., Rücker, B. (2019). Real-Life BPMN (4th edition): Includes an introduction to DMN, Camunda, ISBN: 978-1086302097

Kane, S., Matthias, K. (2018). Docker: Up & Running, 2nd Edition, O'Reilly Media, Inc., ISBN: 9781492036739

Kleppmann, M. (2017). Designing Data-Intensive Applications, O'Reilly Media, Inc., ISBN: 9781449373320

Bellemare, A. (2020). Building Event-Driven Microservices: Leveraging Organizational Data at Scale, O'Reilly Media, ISBN: 978-1492057895

Brikman, Y. (2019), Terraform: Up & Running, 2nd Edition, O'Reilly Media, Inc., ISBN: 9781492046905

Newman, S. (2021). Building Microservices, 2nd Edition. O'Reilly Media, Inc., ISBN: 9781492034025

Gough, J., Bryant, D., Auburn, M. (2023). Mastering API Architecture: Design, Operate, and Evolve API-Based Systems, O'Reilly Media, Inc., ISBN: 9781492090632

The technology that you'll be using

*Business
Processes
execution*



*API
Management*



MicroServices



Messaging

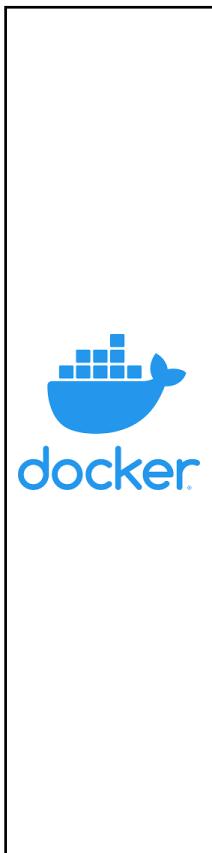


*PaaS
IaaS*

Hyperautomation

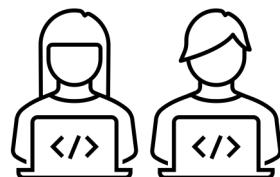


Containerization



Minimum tools to install on your environment

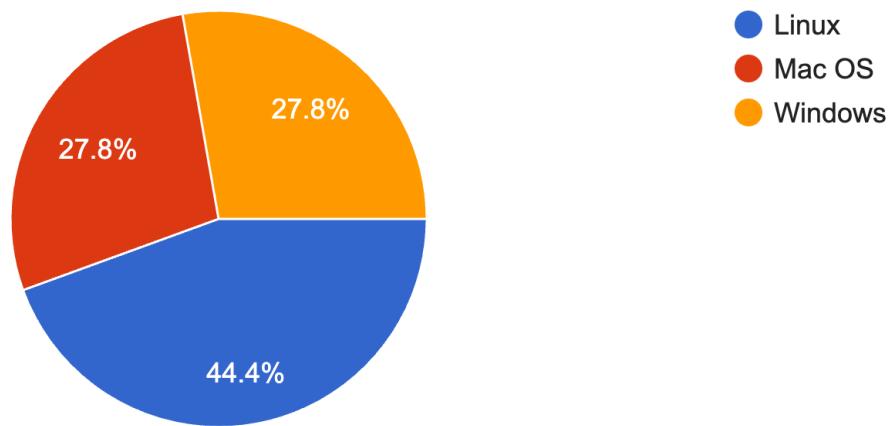
see tutorials for further
details



- Windows: putty + fileZilla
- Linux or macOS: check if ssh and scp are working
- All Operating systems:
 - VS Code <https://code.visualstudio.com/>
 - Eclipse (*optionally to VS Code*) <https://www.eclipse.org/downloads/>
 - JDK 17 <https://adoptium.net/>
 - Maven 3.9.9 <https://maven.apache.org/download.cgi>
 - Terraform <https://developer.hashicorp.com/terraform/downloads>
 - Docker desktop <https://www.docker.com/get-started>
 - Camunda-modeler (v7) <https://camunda.com/download/modeler/>
 - MYQSL Workbench <https://dev.mysql.com/downloads/workbench/>
 - Postman <https://www.postman.com/downloads/>

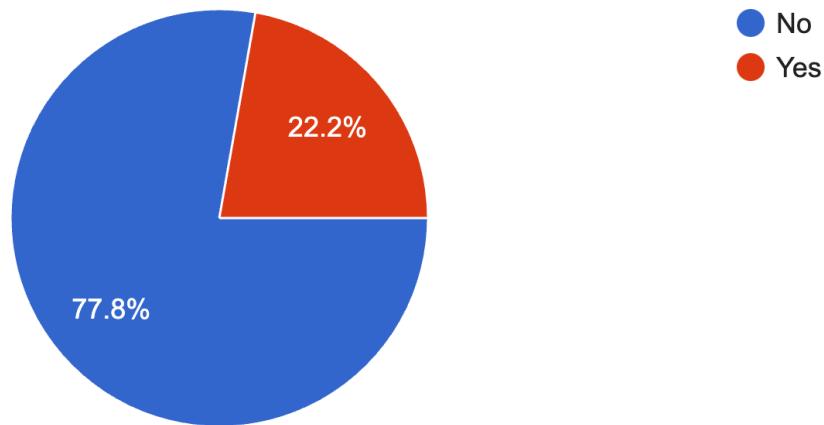
What is the Operating System that you are planning to use during EI course?

18 responses



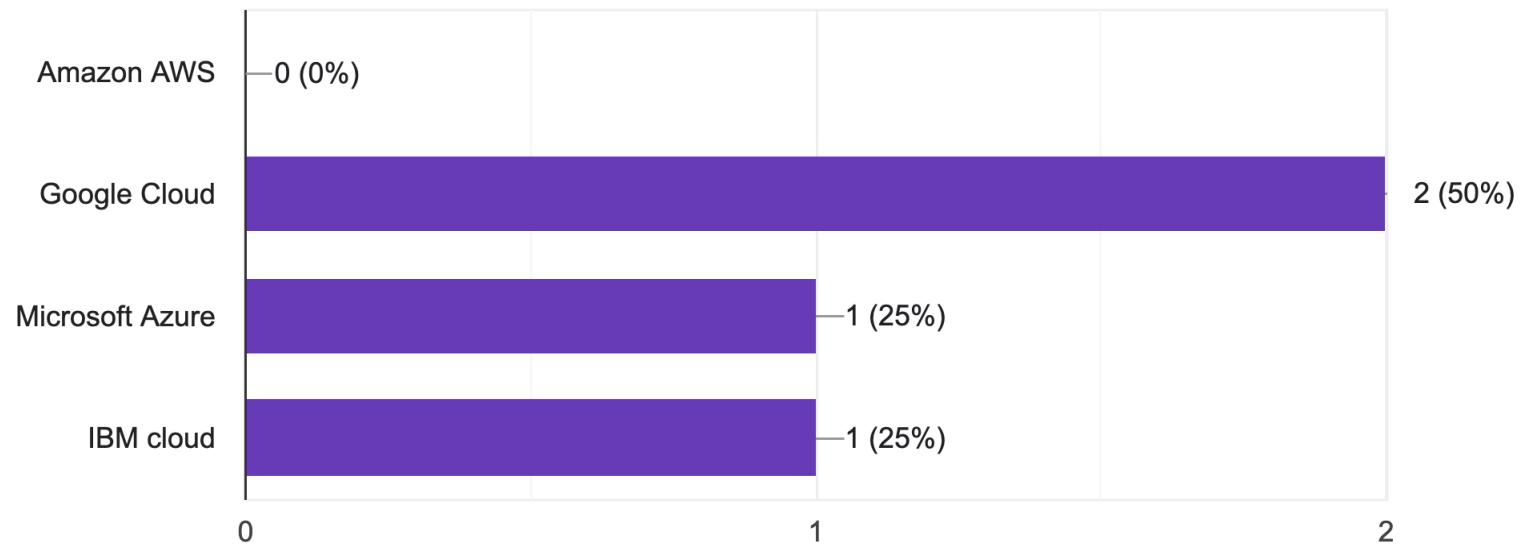
Have you ever developed and/or configured using cloud environments (e.g.: AWS, Microsoft Azure, Google cloud, other....)?

18 responses



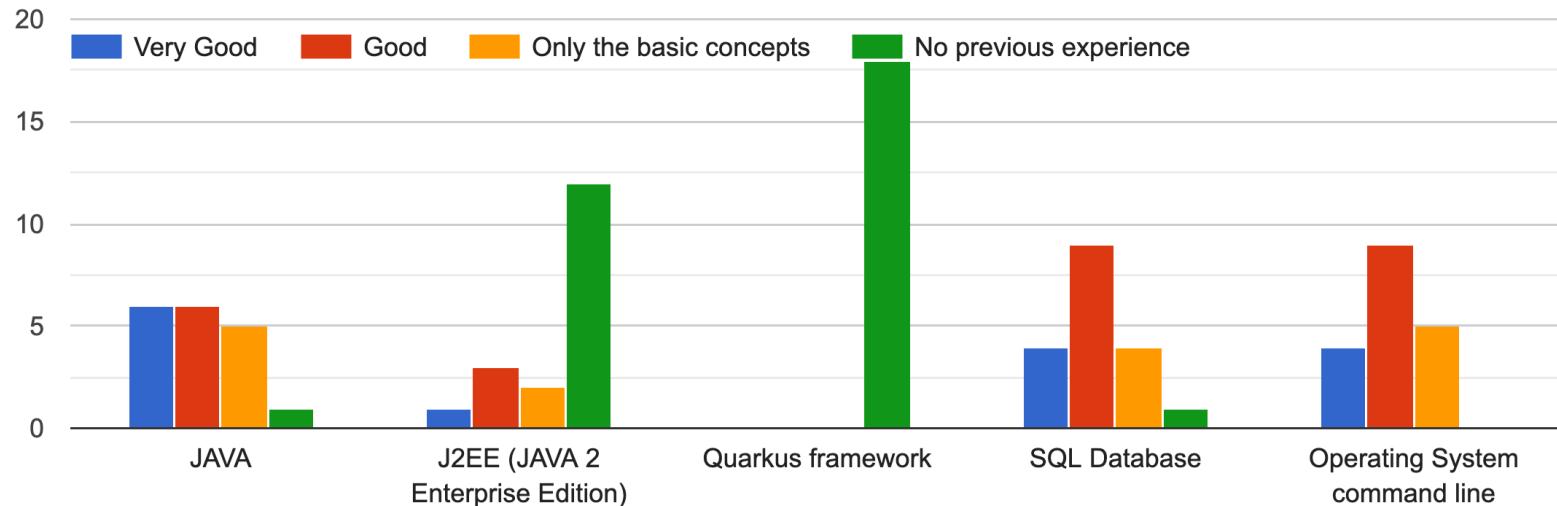
Which cloud environment are you experienced with?

4 responses



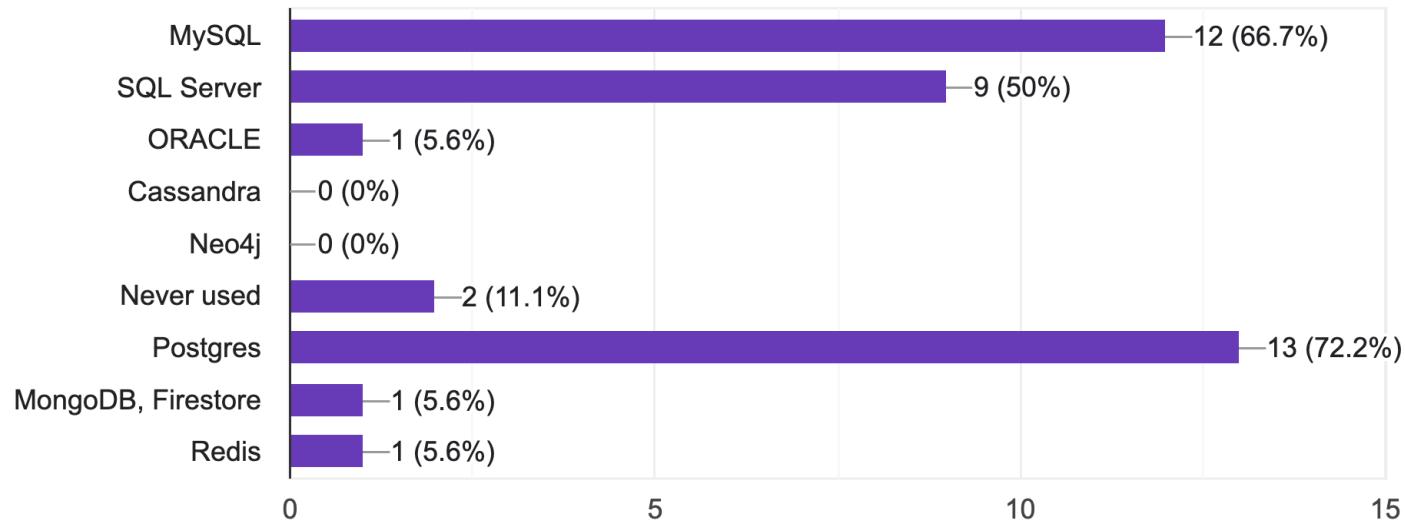
Programming background - What is your experience in the following technologies?

Choose one option per row.



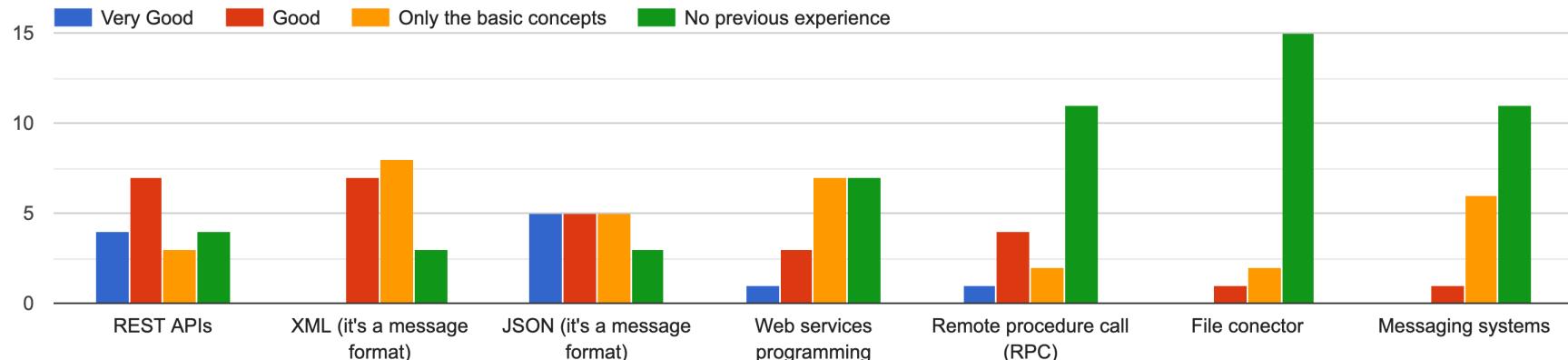
Regarding Database knowledge or experience, which technologies did you use? Choose one or more options.

18 responses



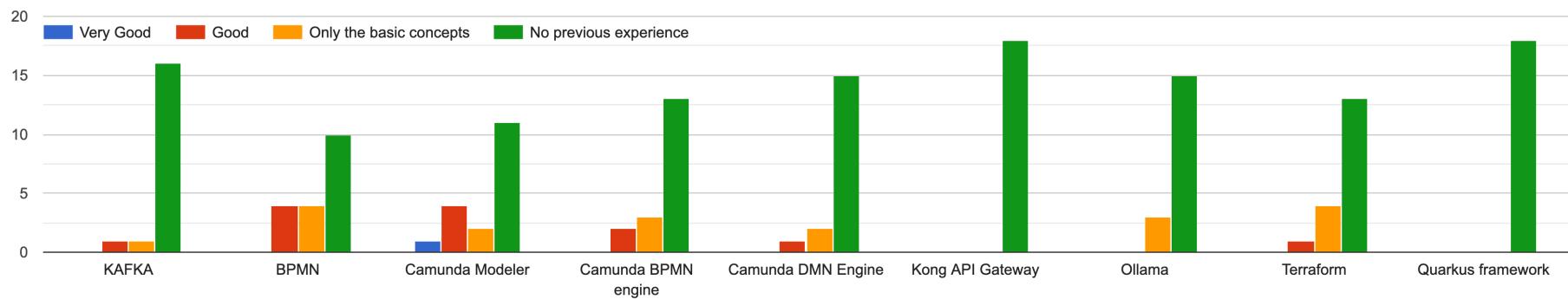
Integration using API - What is your experience in the following technologies?

Choose one option per row.



Knowledge related with other integration systems - What is your experience in the following technologies?

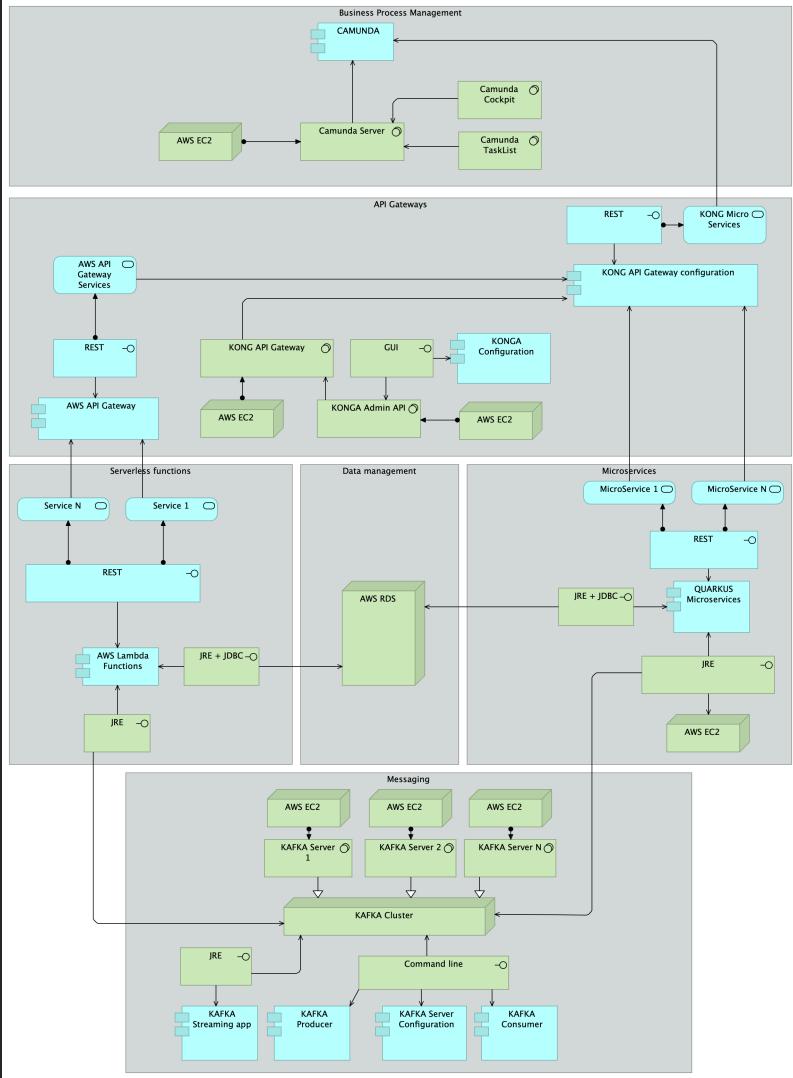
Choose one option per row.





IE Technology Stack

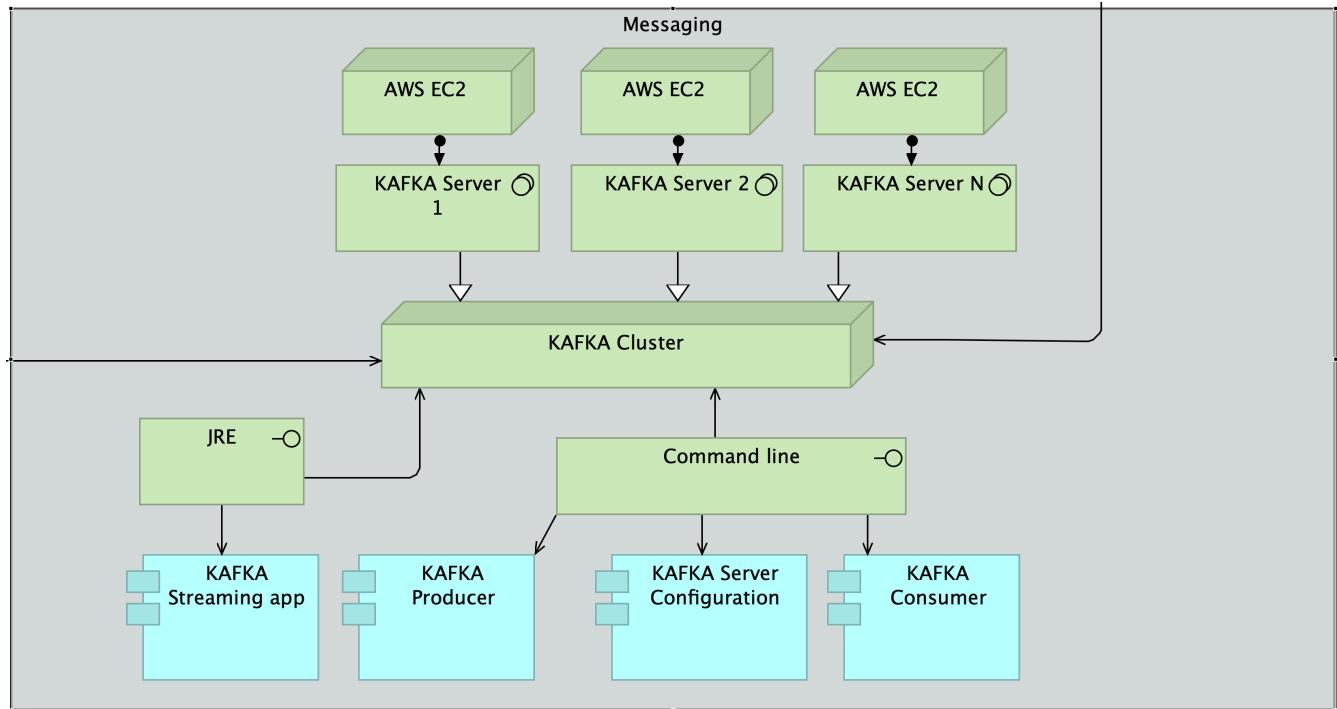
Overall view





IE Technology Stack

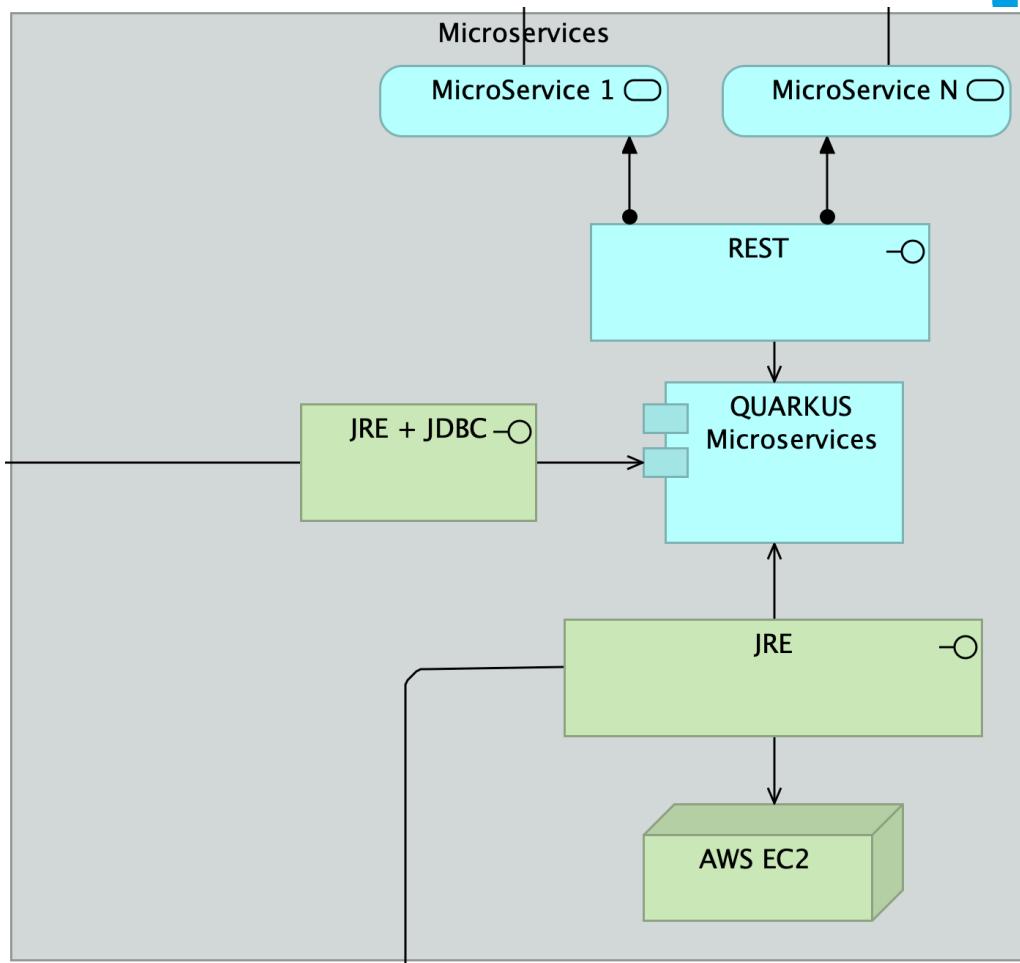
Messaging layer





IE Technology Stack

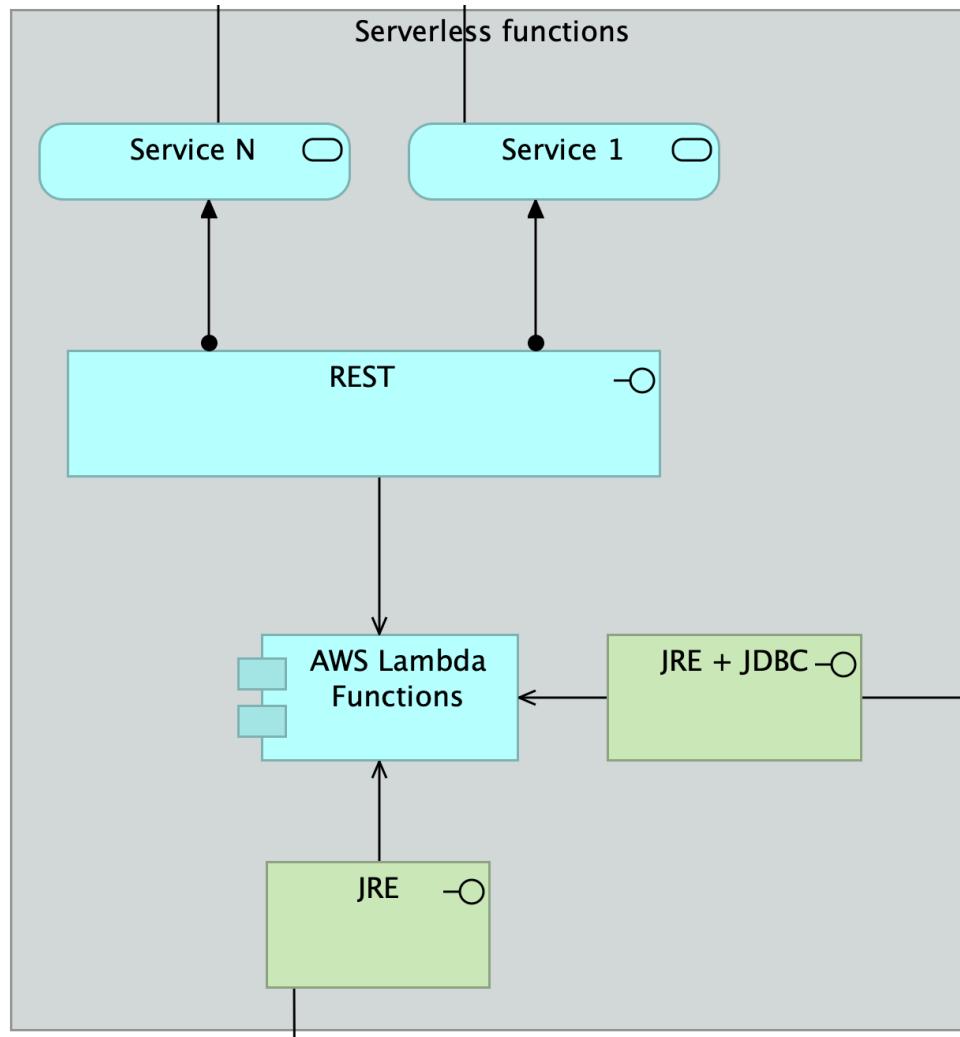
MicroService layer





IE Technology Stack

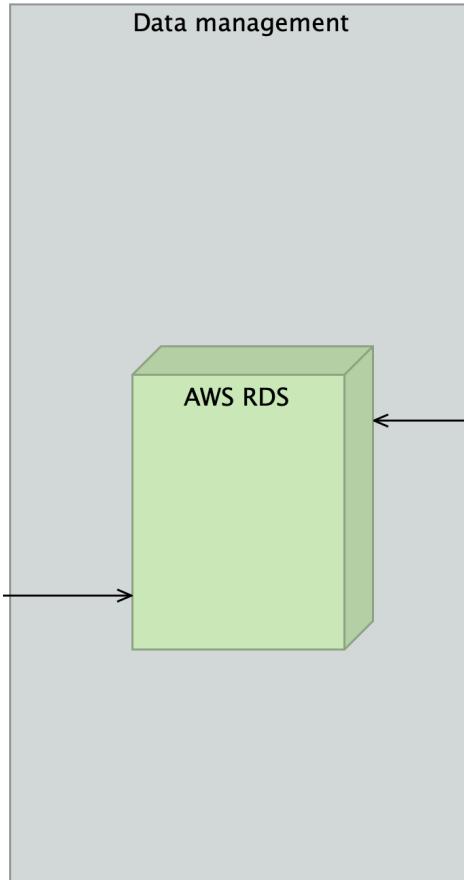
Serverless layer

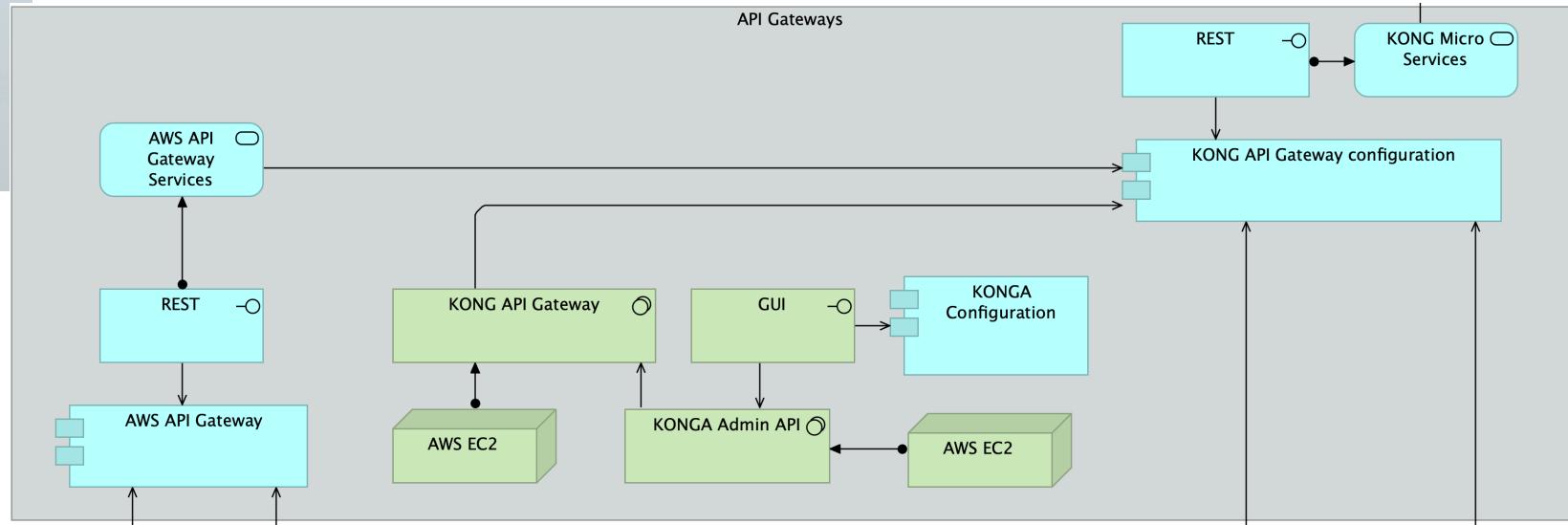




IE Technology Stack

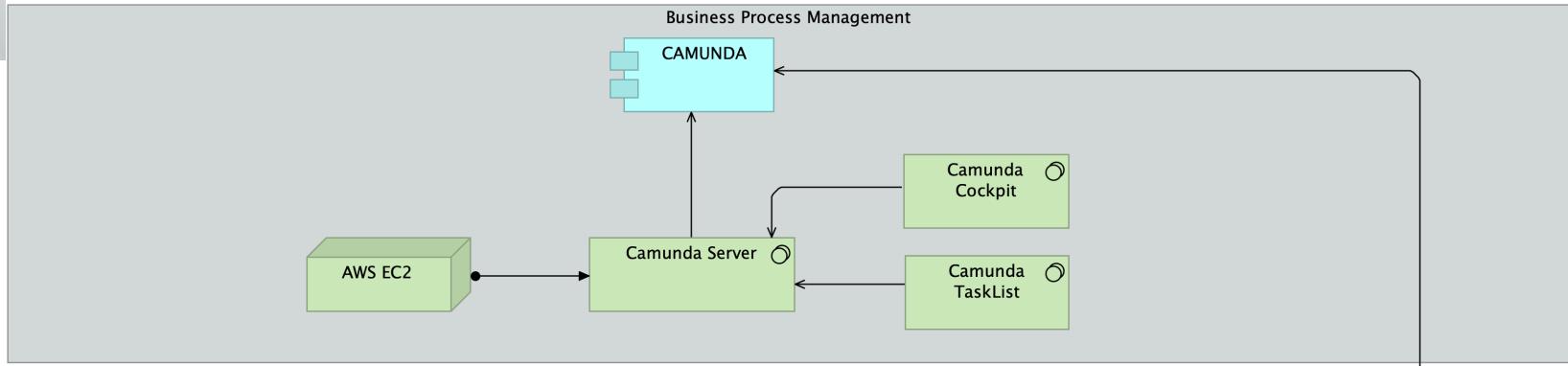
Data management layer





IE Technology Stack

API Gateways layer



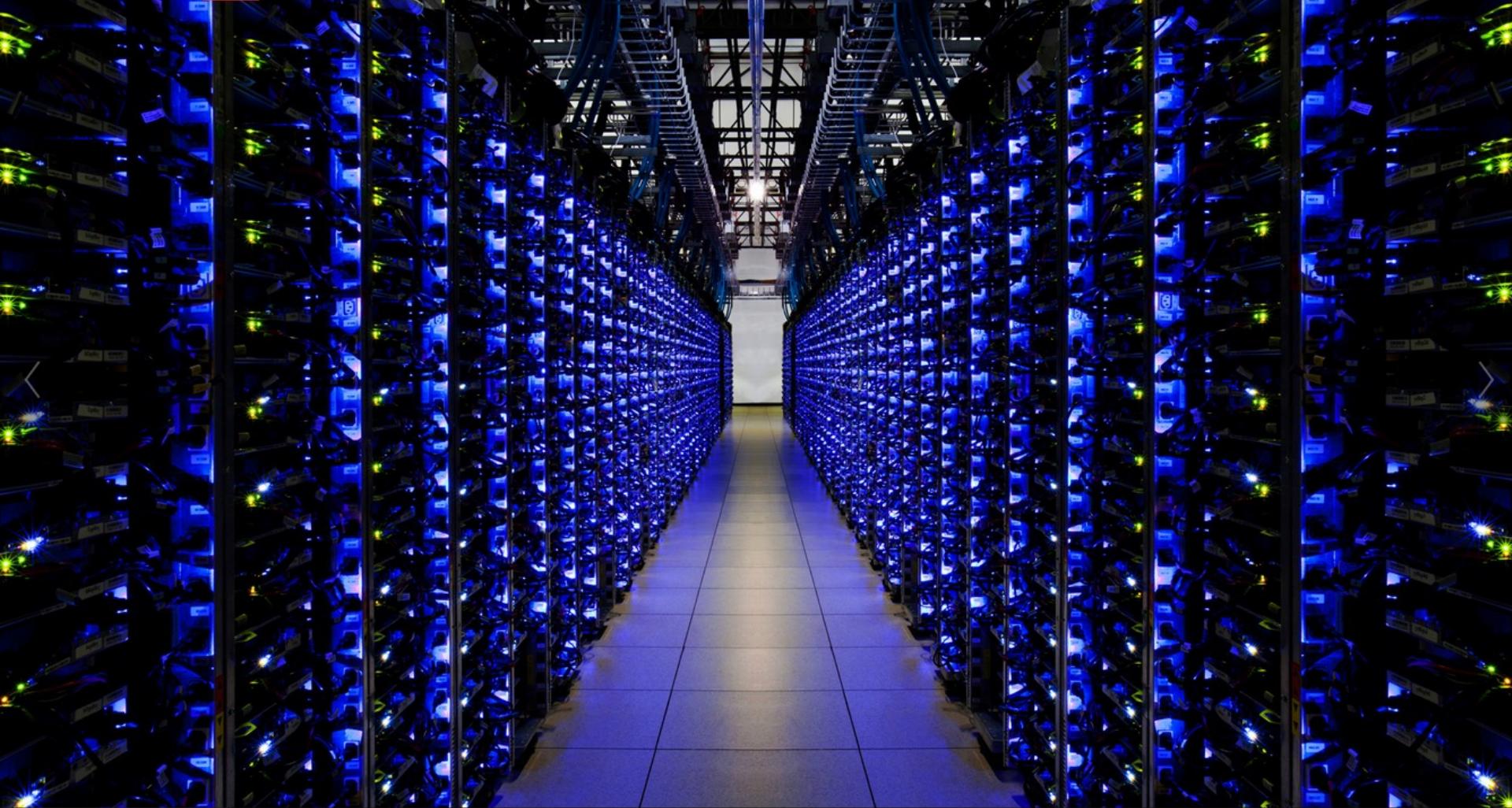
IE Technology Stack

Business Process management layer

Digital economy

Why is enterprise integration needed?





From: <http://www.google.com/about/datacenters/>

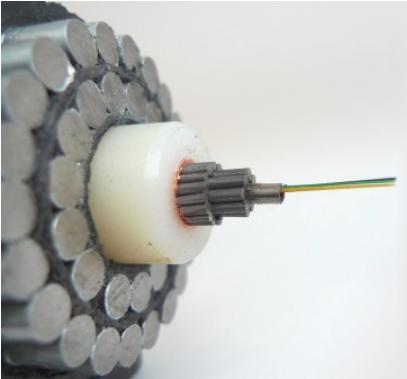
The internet's undersea world

In vast majority of the world, communications are not carried by satellites but by altogether older oceans. As a ship accidentally wipes out Asia's cable, we rely on millions of wires of less than 10cm diameter to link us all together.

Fibre-optic submarine cable systems

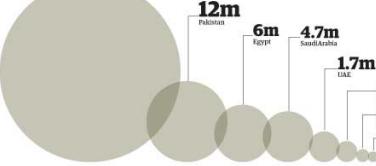
In-service
Planned
Damaged

Cables carry a mixture of capacity and bandwidth. Some are new construction or working pre-war 1945.



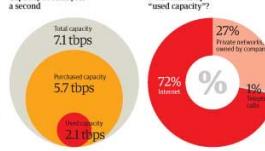
Internet users affected by the Alexandria accident
The main countries affected in Wednesday's event

60m
India



World cable capacity
Some companies lease part of their space on other systems to sell bandwidth to other carriers. Carries have their own private bandwidth reserves. On the trans-Atlantic route 80% of the bandwidth is purchased, but only 22% is used.

Capacity in terabytes a second



The longest submarine cables
The longest system is from Norway to Germany to Italy. South Korea connects 32 different countries with 29 landing points.

Seabone-1	39,000 km
Southern Cross	30,500 km
China-1	30,475 km
FLAG Europe-Asia	29,000 km
South America-1	25,000 km

The world's cables in bandwidth
The trans-Atlantic cable system connects North America to Europe in a single fiber pair with a capacity of 640,000 bytes per second. Since then, total trans-Atlantic cable capacity has soared to over 2 trillion bytes.

Total capacity	7.1 tbps
Per fiber capacity	5.7 tbps
Per fiber bandwidth	2.1 tbps



Estimated international bandwidth usage by country



Cross-section of a cable
Cables are typically 60 mm in diameter and weigh 10,000 kilograms a kilometer. In deeper waters, thicker and less insulated cables are used.

Gathering fibers
Tin coated copper wire
Optic fibers
Silica gel

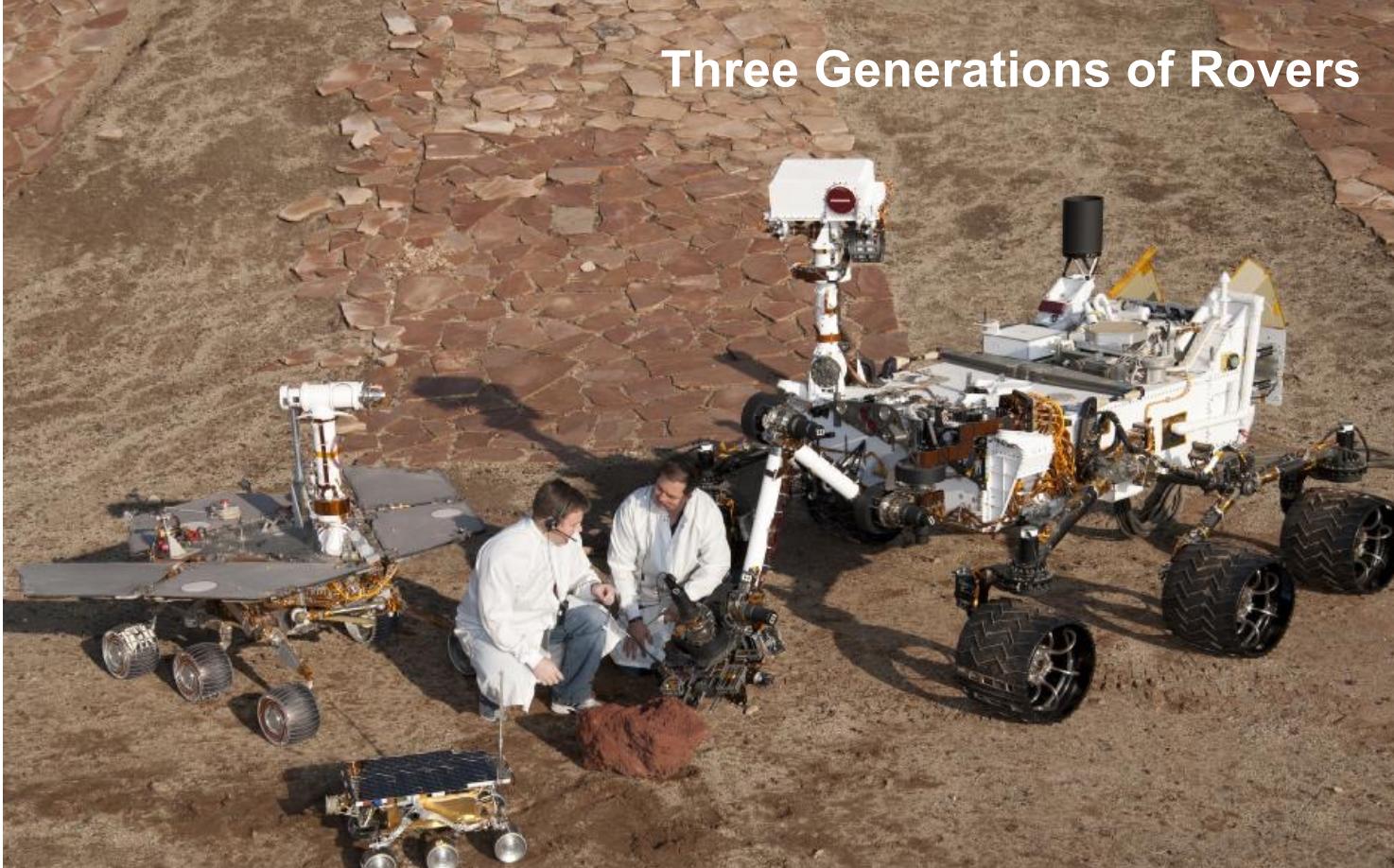
Buffering material (plastic/lead)
Ultra-thin strength fiber
Copper sheath



Combining data + processing + communications



Three Generations of Rovers



Front and center is the flight spare for the first Mars rover, Sojourner, which landed on Mars in 1997 as part of the Mars Pathfinder Project. On the left is a Mars Exploration Rover Project test rover that is a working sibling to Spirit and Opportunity, which landed on Mars in 2004. On the right is a Mars Science Laboratory test rover the size of that project's Mars rover, Curiosity, which is on course for landing on Mars in August 2012.

The Future of Jobs

FIGURE 22

Top 20 job roles in increasing and decreasing demand across industries

↗ Increasing demand

1	Data Analysts and Scientists
2	AI and Machine Learning Specialists
3	Big Data Specialists
4	Digital Marketing and Strategy Specialists
5	Process Automation Specialists
6	Business Development Professionals
7	Digital Transformation Specialists
8	Information Security Analysts
9	Software and Applications Developers
10	Internet of Things Specialists
11	Project Managers
12	Business Services and Administration Managers
13	Database and Network Professionals
14	Robotics Engineers
15	Strategic Advisors
16	Management and Organization Analysts
17	FinTech Engineers
18	Mechanics and Machinery Repairers
19	Organizational Development Specialists
20	Risk Management Specialists

↘ Decreasing demand

1	Data Entry Clerks
2	Administrative and Executive Secretaries
3	Accounting, Bookkeeping and Payroll Clerks
4	Accountants and Auditors
5	Assembly and Factory Workers
6	Business Services and Administration Managers
7	Client Information and Customer Service Workers
8	General and Operations Managers
9	Mechanics and Machinery Repairers
10	Material-Recording and Stock-Keeping Clerks
11	Financial Analysts
12	Postal Service Clerks
13	Sales Rep., Wholesale and Manuf., Tech. and Sci.Products
14	Relationship Managers
15	Bank Tellers and Related Clerks
16	Door-To-Door Sales, News and Street Vendors
17	Electronics and Telecoms Installers and Repairers
18	Human Resources Specialists
19	Training and Development Specialists
20	Construction Laborers

Source

Future of Jobs Survey 2020, World Economic Forum.

FIGURE 23 | Emerging roles clustered into the jobs of tomorrow

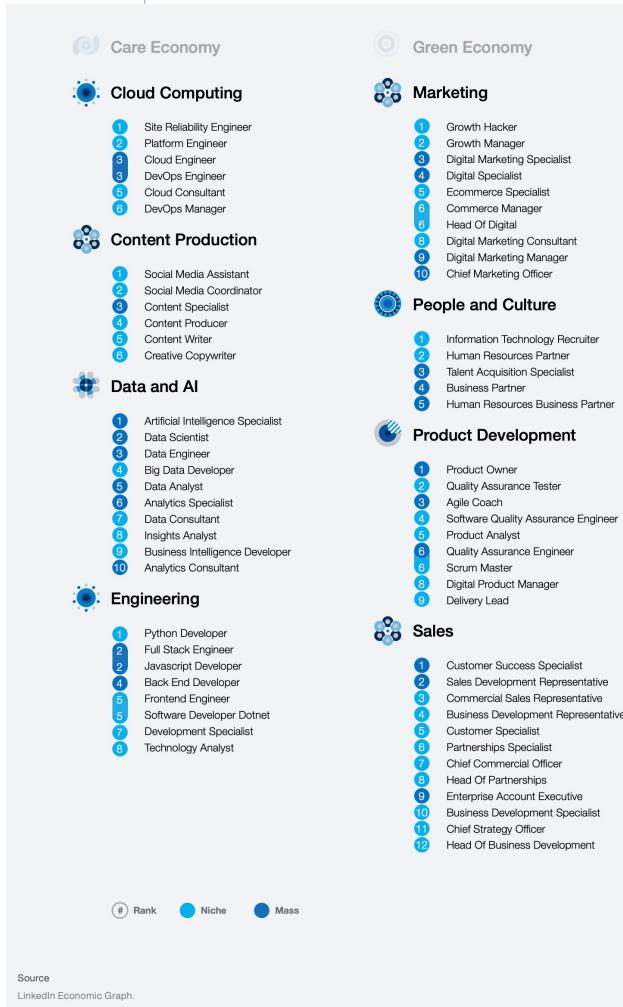
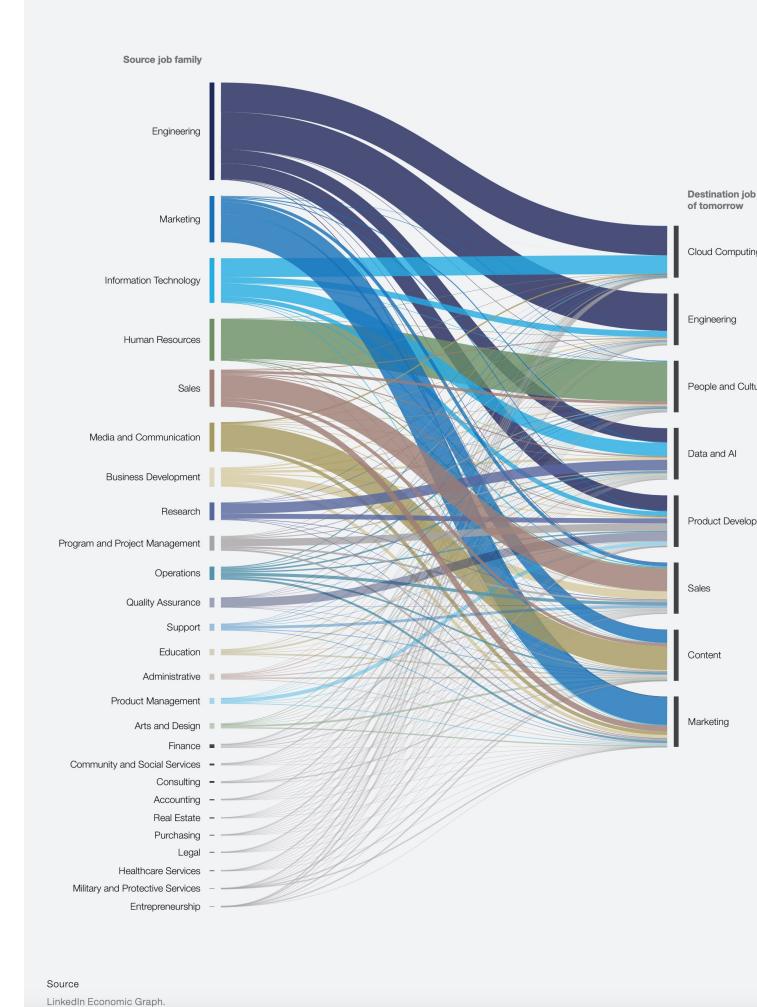


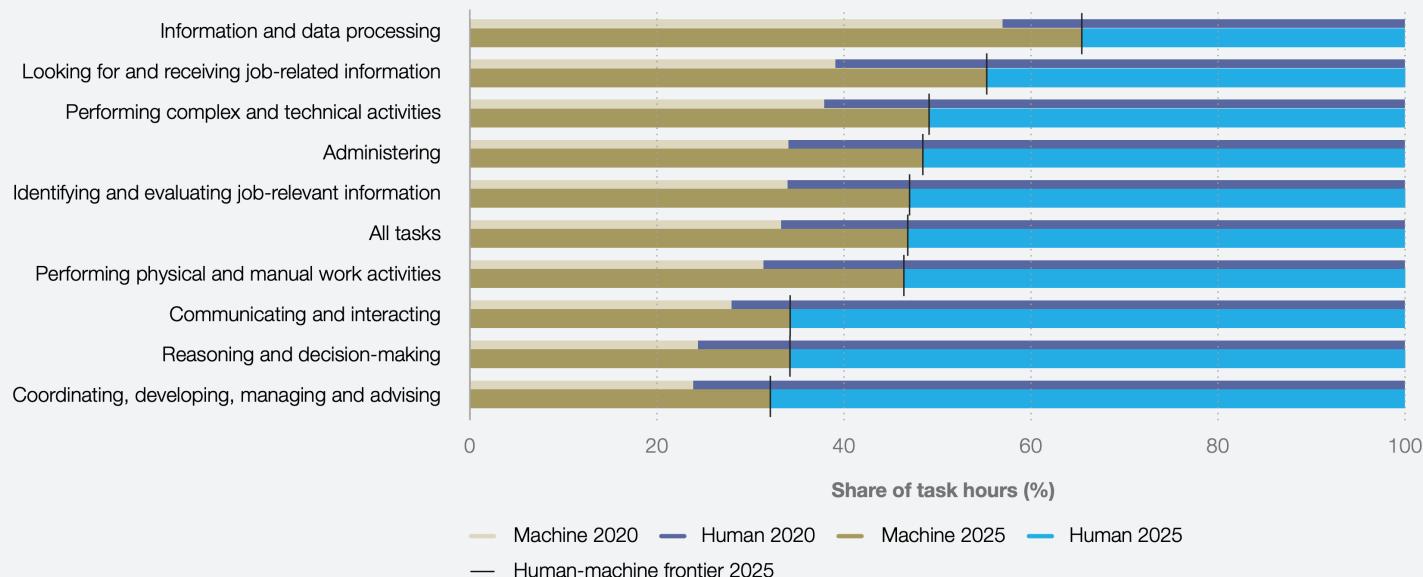
FIGURE 25 | Transitions into the jobs of the future



Machines taking working hours....

FIGURE 21

Share of tasks performed by humans vs machines, 2020 and 2025 (expected), by share of companies surveyed

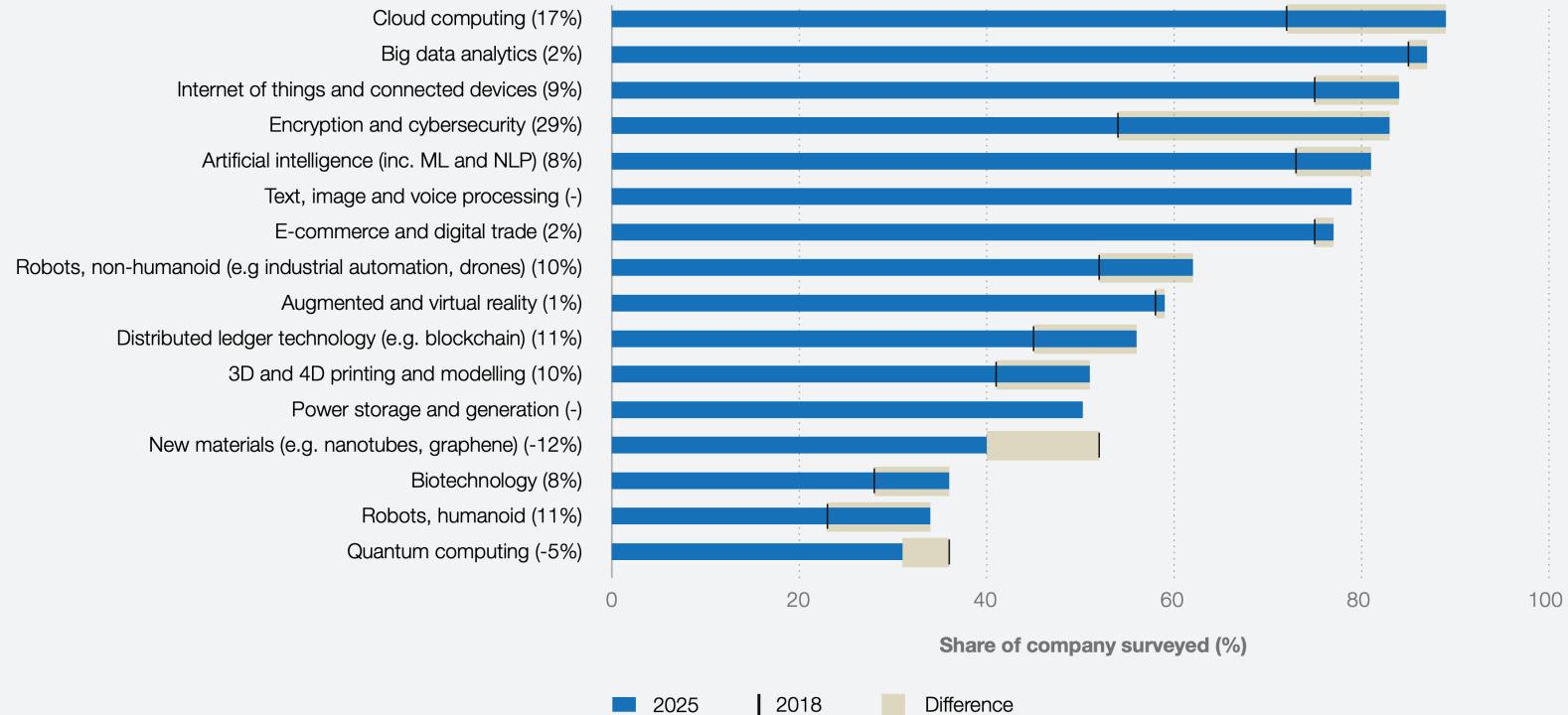


Source

Future of Jobs Survey 2020, World Economic Forum.

FIGURE 18

Technologies likely to be adopted by 2025 (by share of companies surveyed)



Source

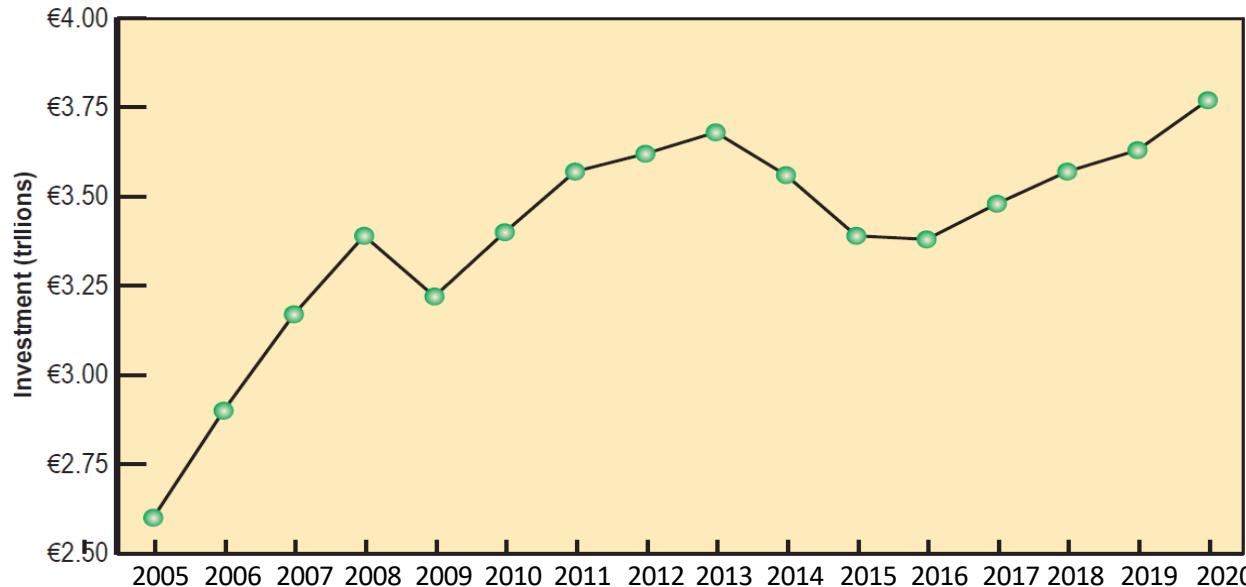
Future of Jobs Survey 2020, World Economic Forum.

Digital Transformation

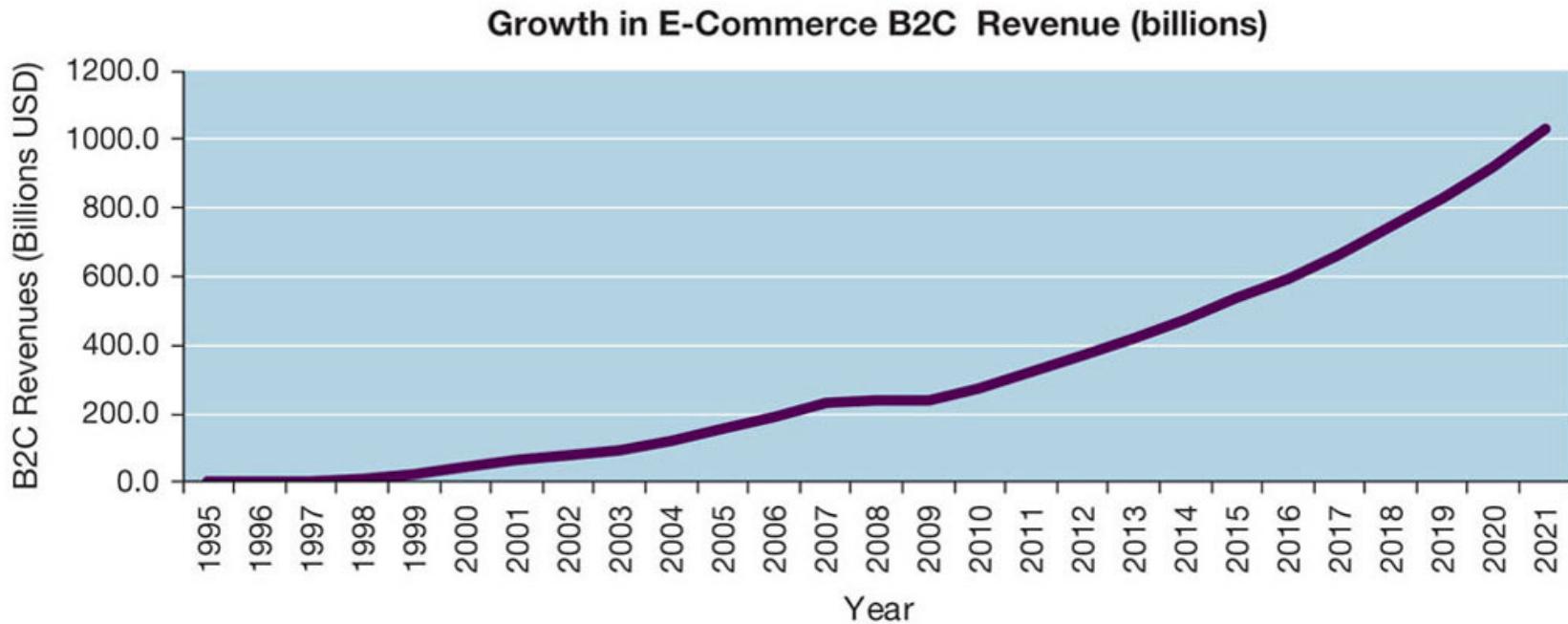


INFORMATION TECHNOLOGY CAPITAL INVESTMENT

IT investment now accounts for an estimated 20% of all capital investment (Laudon & Laudon, 2017)



The Growth of E-Commerce



Digital landscape everywhere



Traditional goods exchanged using digital markets - *Amazon warehouse*





Goods *versus* Digital Goods

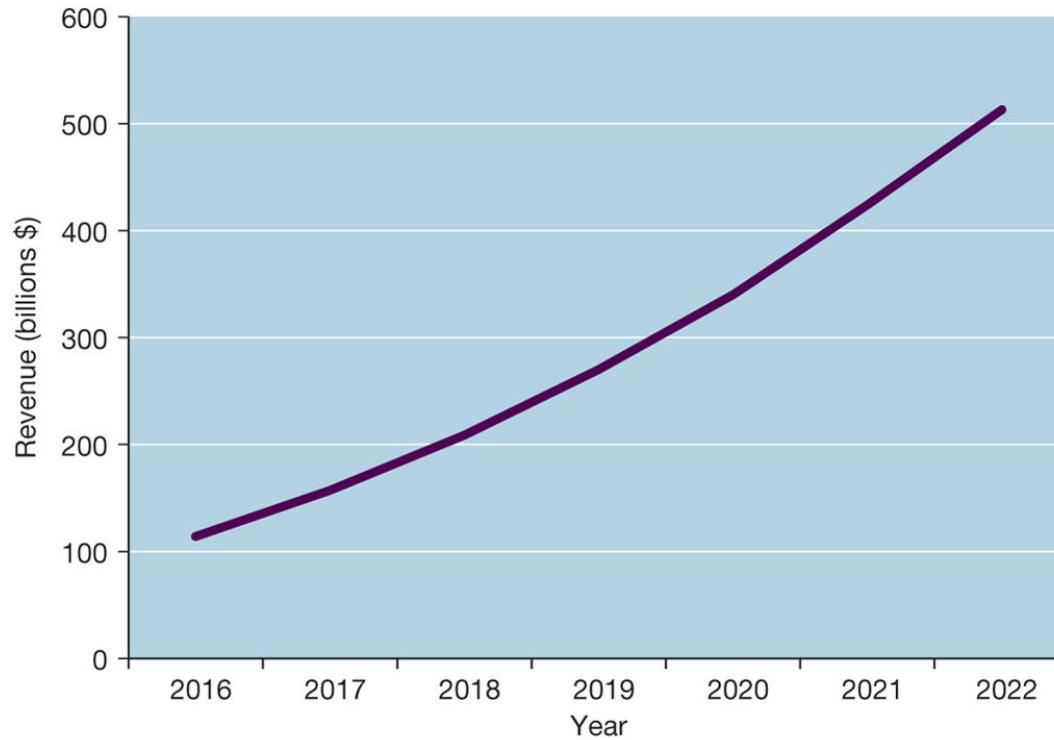
...

Books
Software
Newspapers
Magazines
.....etc.



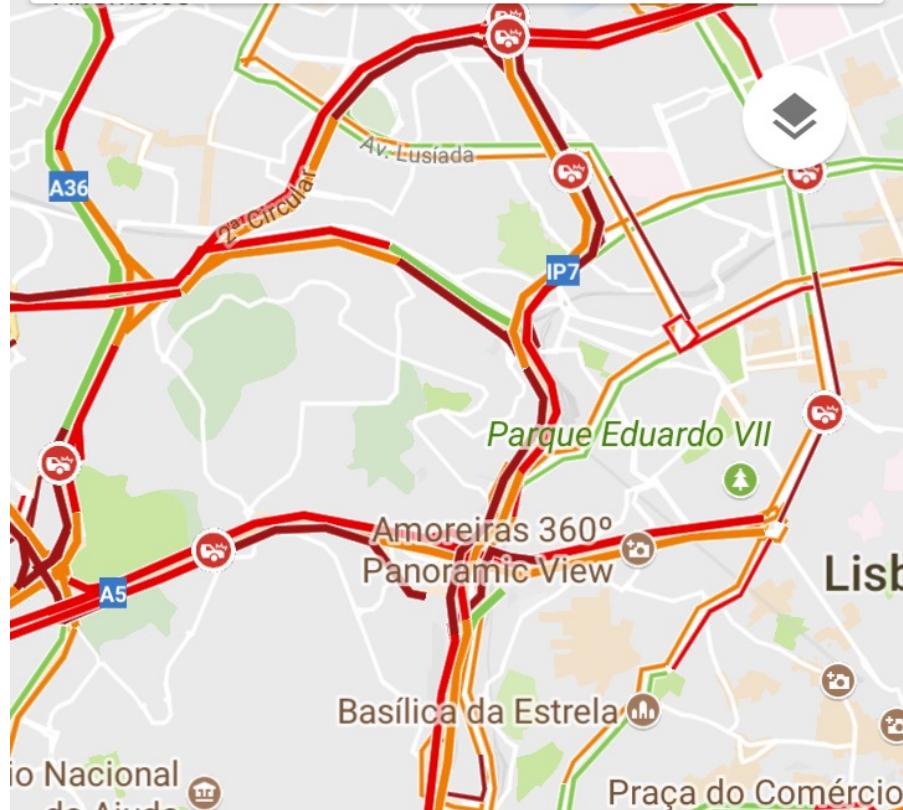
mixing things –
digital markets using
traditional
environments

Mobile Retail Commerce Revenues



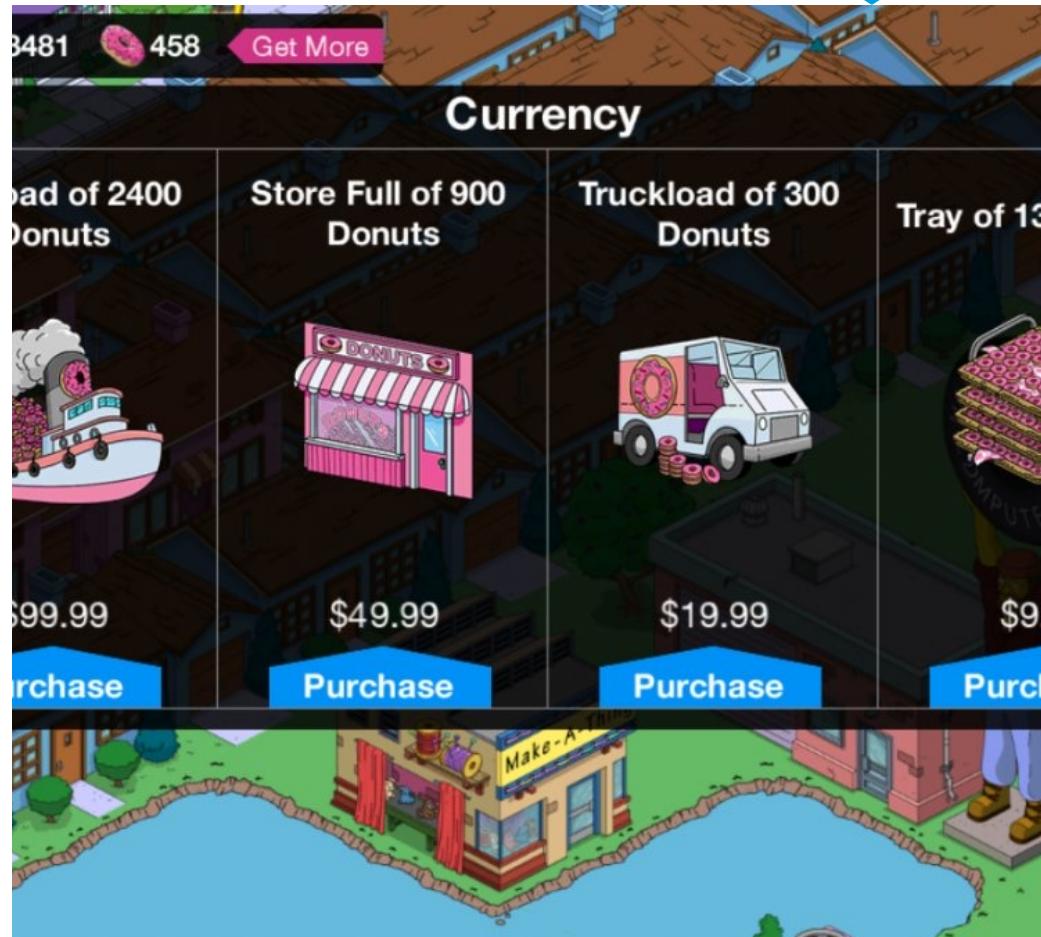
Mobile environments

Location based services and applications



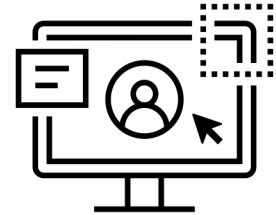
Mobile environments

mobile games
micro-payments



Enterprise Systems Applications

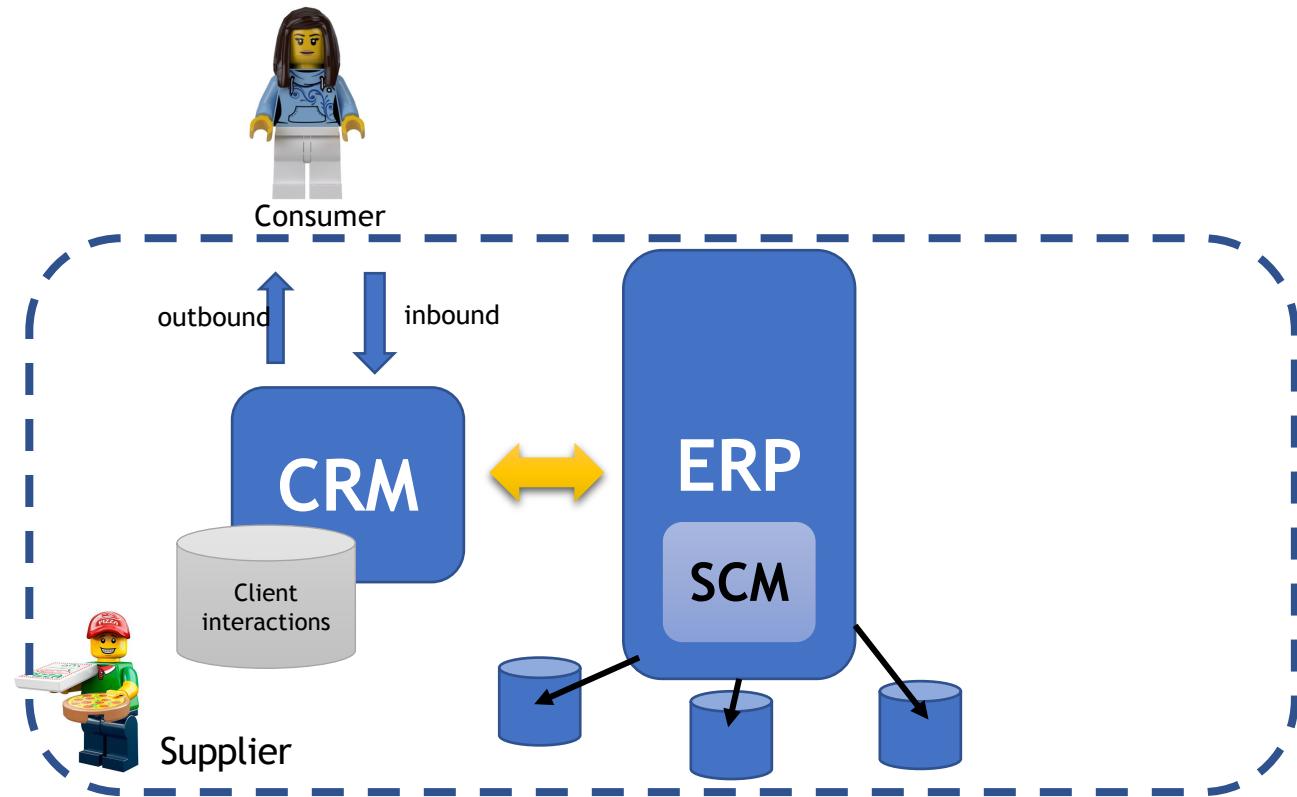
How is a digital enterprise designed?



Enterprise Information Systems (EISs)

Can be defined as “software systems for business management, encompassing modules supporting organisational functional areas such as planning, manufacturing, sales, marketing, distribution, accounting, financial, human resources management, project management, inventory management, service and maintenance, transportation and e-business”.

They are made of computers, software, people, processes and data.



Supply Chain Operating supported by TPS ([Transaction processing systems](#))

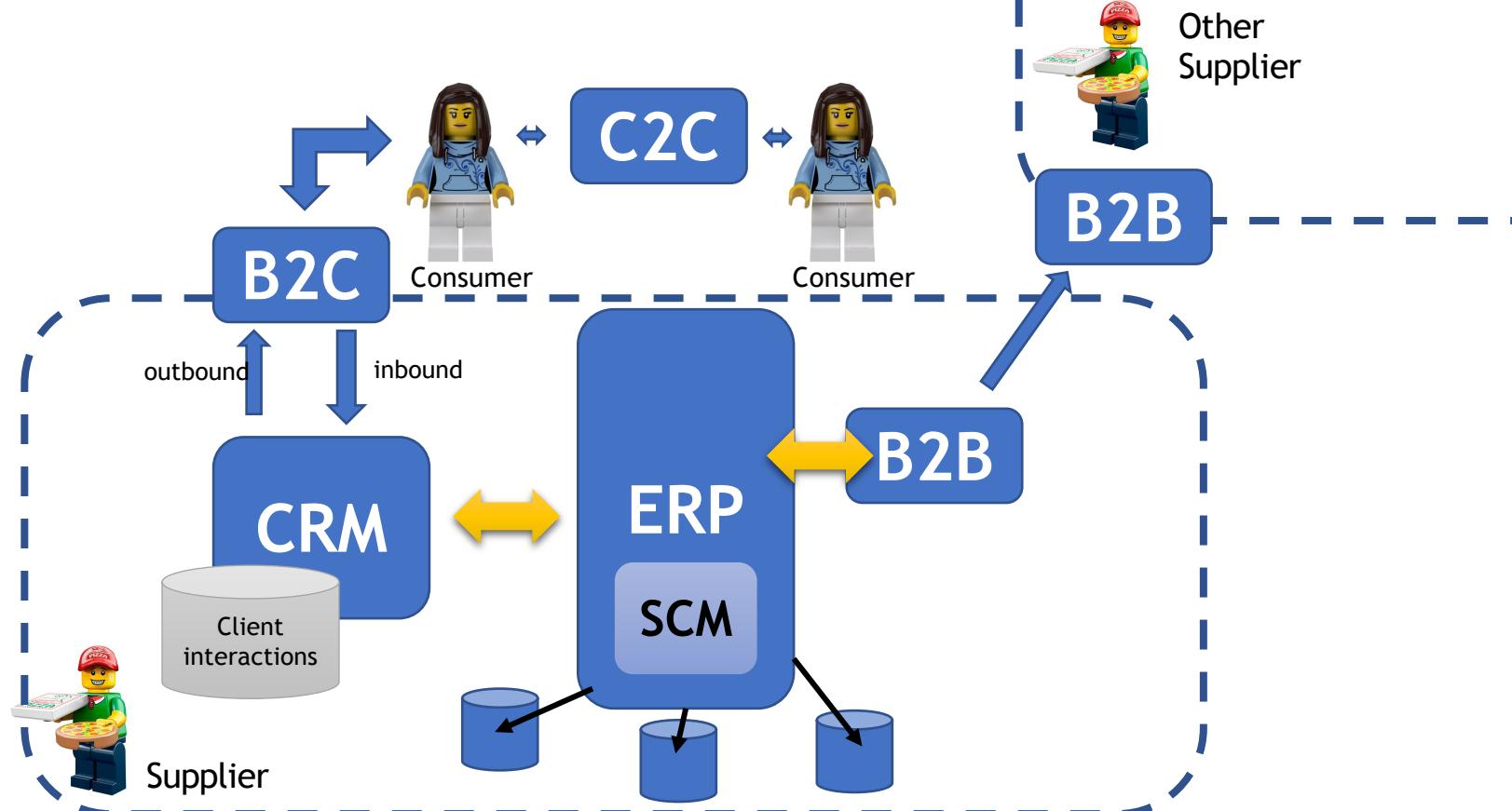
Customer Relationship Management Systems

- Knowing the customer
 - In large businesses, too many customers and too many ways customers interact with firm
- Customer relationship management (CRM) systems
 - Capture and integrate customer data from all over the organization
 - Consolidate and analyze customer data
 - Distribute customer information to various systems and customer touch points across enterprise
 - Provide single enterprise view of customers

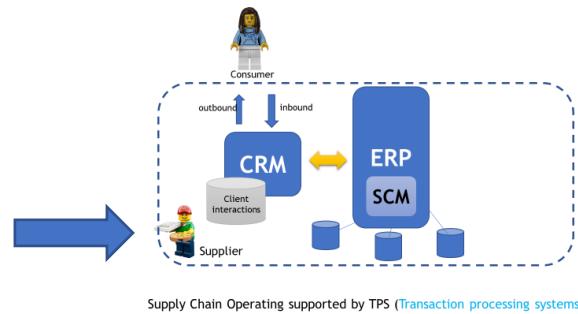
FIGURE 9.6 CUSTOMER RELATIONSHIP MANAGEMENT (CRM)



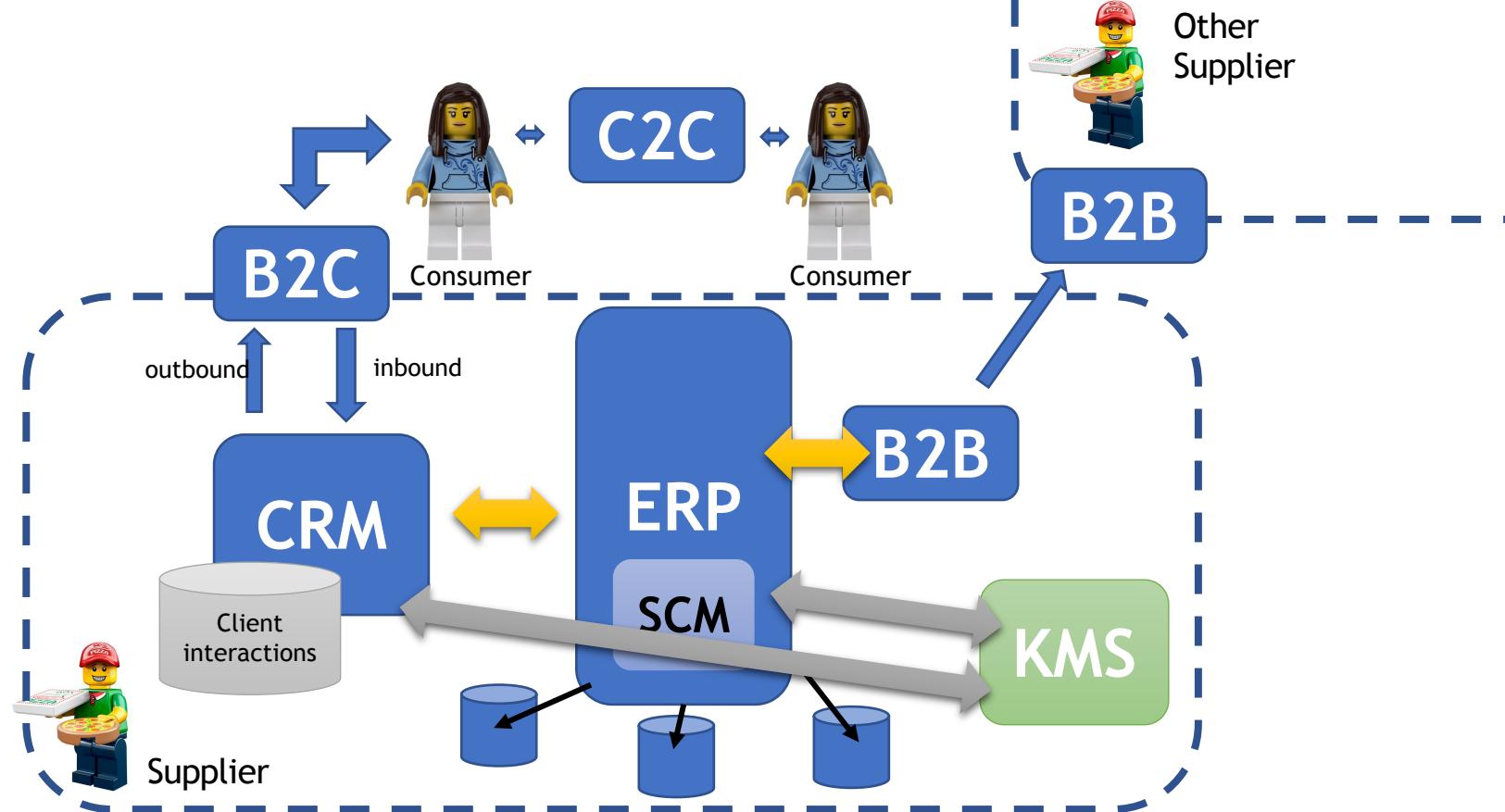
CRM systems examine customers from a multifaceted perspective. These systems use a set of integrated applications to address all aspects of the customer relationship, including customer service, sales, and marketing.



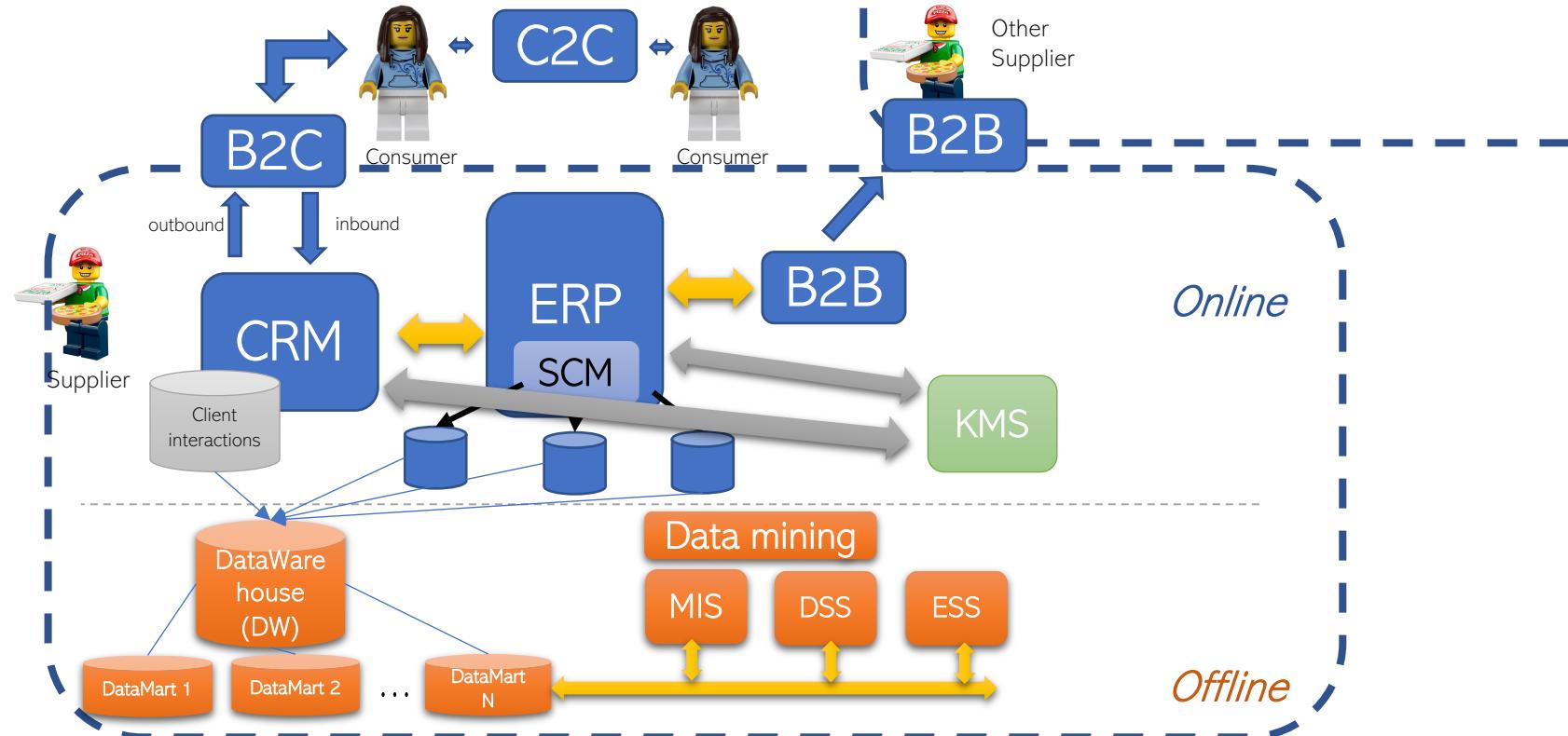
Supply chain management systems SCMs



- Push-based model (build-to-stock)
 - Schedules based on best guesses of demand
- Pull-based model (demand-driven)
 - Customer orders trigger events in supply chain
- Sequential supply chains
 - Information and materials flow sequentially from company to company
- Concurrent supply chains
 - Information flows in many directions simultaneously among members of a supply chain network



Supply Chain Operating supported by TPS (Transaction processing systems)

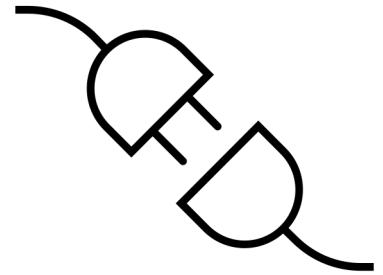


The evolution of EISs

Table 1

Enterprise information systems generations (extended from [4]).

EIS Generation	1	2	3	4	5	6
	Application centric	Data centric	Process centric	Human centric	Things centric	Everything centric
Timeline	Around 80's	Around 90's	Around 00's	Around 10's	Around 15's	Around 20's
Iconic technology	MRP	ERP	BPM	HMI	Sensors	Devices
Technology drivers	Databases	Dbms, client-server architecture	Internet, soa	Semantic networks, intelligent social media, Cloud computing,	Smart things, wireless sensor networks, Big data,	Real plug and play systems, open and trusted Platforms, trustworthy infrastructures, interoperability service utility
				Virtual and augmented reality	Service science, cloud computing	
Deployment environment	Local systems		Digital systems		Cyber-physical systems	
Business challenge	Efficacy	Efficiency	Effectiveness	Resilience	Sensitiveness	Proactiveness
Organisational challenge	Support of departments	Support of enterprises	Support of supply chains	Support of social networks	Support of sensor networks	Support of interplay networks
Technology challenge	Systems integration		Systems interoperability			
Knowledge challenge	Structured data	Integrated data	Dynamic data	Real-time data		Inferred data



Enterprise Systems Integration & Interoperability

What is integration?

The integration term appears very often in the literature, *i.e.*, books, journals, conferences, newspapers...etc., yet, it has evolved throughout the years, namely:

- the earliest texts referred to “**System integration**”,
- afterwards it was called “**Application Integration**”
- more recently “**Enterprise integration**” and “**Enterprise interoperability**”
- And now often described as “**Enterprise cloud integration**”

Enterprise Integration *def.*?

- Is the process of ensuring the interaction between enterprise entities necessary to achieve domain objectives
- Can be approached in various manners and at various levels, e.g., (i) physical integration, (ii) application integration, (iii) business integration, (iv) through enterprise modeling, and (v) as methodological approach to achieve consistent enterprise-wide decision-making.

Enterprise Systems Integration

Available knowledge base and trends

Results for "enterprise integration" (All Fields)

from Web of Science, Google Scholar, IEEE, ACM on 17.1.2023

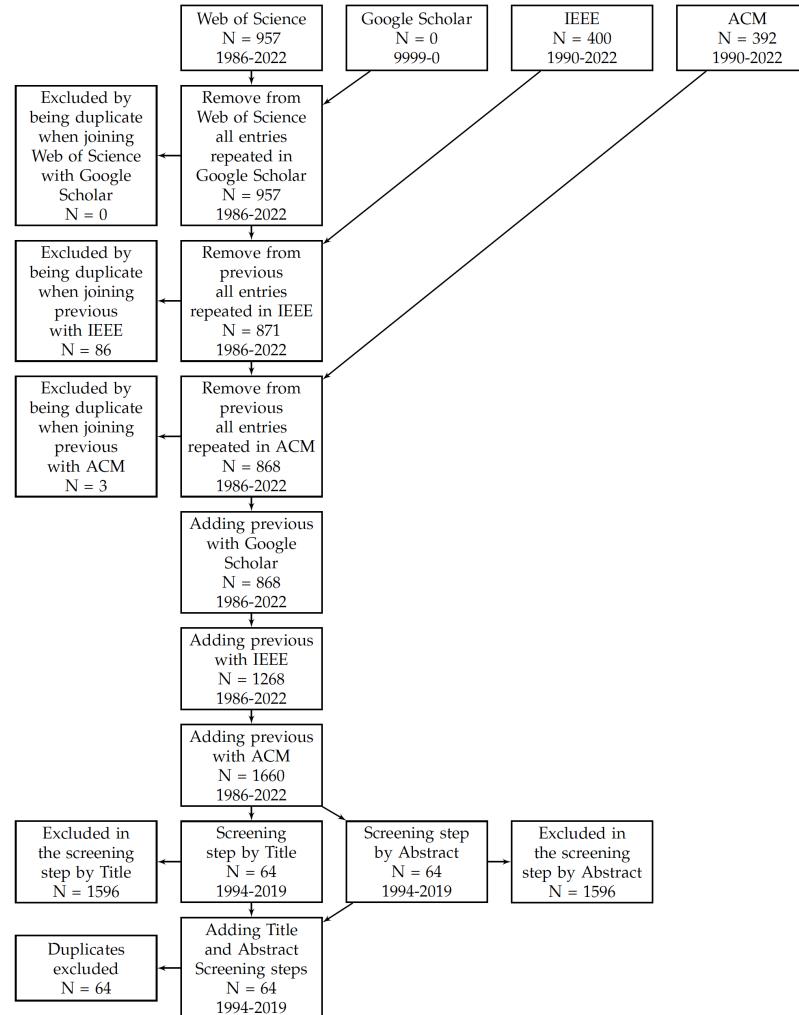


Fig. 1: The refinement steps executed in this SLR procedure and the resulting number of articles.

Results for "enterprise integration" (All Fields) from Web of Science, Google Scholar, IEEE, ACM on 17.1.2023

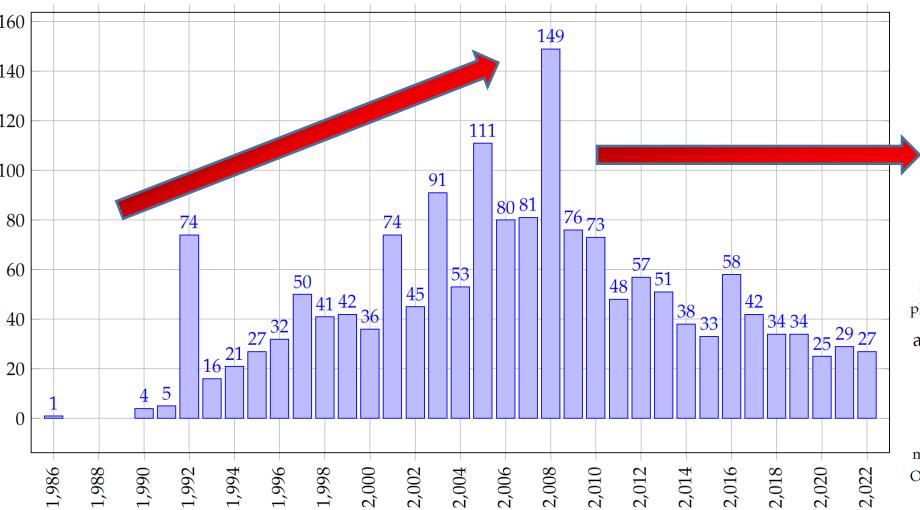


Fig. 4: Distribution of the documents by years. 1660 papers were considered.

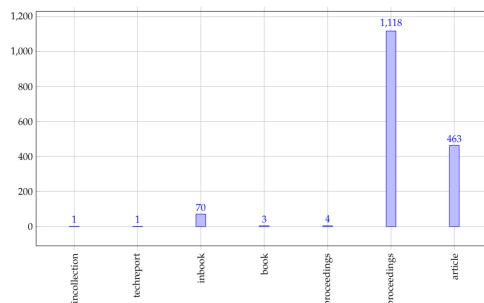


Fig. 2: Wordcloud of the main concepts extracted from Title.

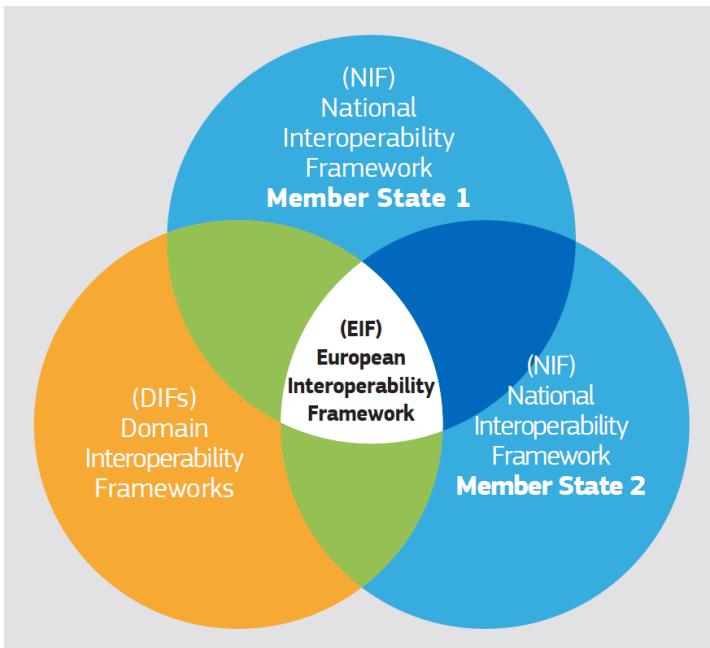
mergence multi-view developments metadata modeling Extended focus improved problem Special manufacturing model preliminary plan Integrated modelling-based based Agent Synthesising issue Capability methodologies collaborative Coordinates impact hierarchy profile analysis classification . cooperative standards Studies Project 1 Web self-controlling integration processing agile Toward theory Inventory ERP scheme independent enterprise global roadmap Addressing into computer systems Comparison after acquisition resilience business Enabler GERM Market-driven Annotation framework future organization metamodels Has through New habitual strength Semantic networking language Knowledge reference assessment view bridging ontologies Object-oriented ontology Boeing enactment markup resource processes? domains interoperability engineering CIMOSA tactical traceability users SQL volume UML principles Service-Oriented oriented alignment Steps PERA no Architectural virtual distance knowledge-based challenges proactive Information constructs representation network ecology provenance Opening Quality proposition Introducing Features supporting value process motion implementation cross Service Data Change generic Guidance Grand industrial performance eMarketplaces perspectives management infrastructure investigation compliant risk strategy maintenance applications

Enterprise Interoperability *def.?*

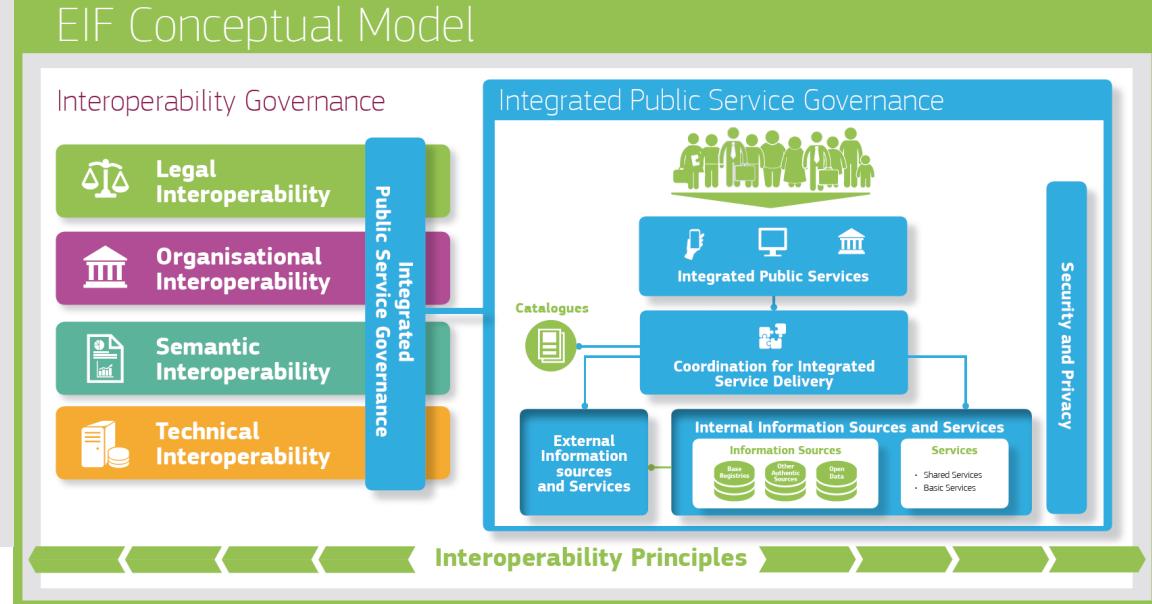
- Is the ability for two systems to understand one another and to use functionality of one another
- "*inter-operate*" implies that one system performs an operation for another system
- In the context of networked enterprises, interoperability refers to the ability of interactions (Exchange of information and services) between enterprise systems
- Interoperability is considered as significant if the interactions can take place at least on three different levels: **data**, **services** and **processes**, with a semantics defined in each business context

ISA² - Interoperability solutions for public administrations, businesses and citizens

https://ec.europa.eu/isa2/eif_en/



EIF Conceptual Model



ISA² - Interoperability solutions for public administrations, businesses and citizens

https://ec.europa.eu/isa2/eif_en/

4 LAYERS



The truck width does not match



EIF promotes the sharing and reuse of common infrastructures, services and IT systems



The two wagons do not match



EIF calls upon public administrations to structure their data in commonly agreed formats



Complex intersection with red lights in all directions.



EIF encourages public administrations to simplify their organisation, streamline their processes and listen to the needs of business and citizens



The distance travelled in a straight line is considerably shorter.



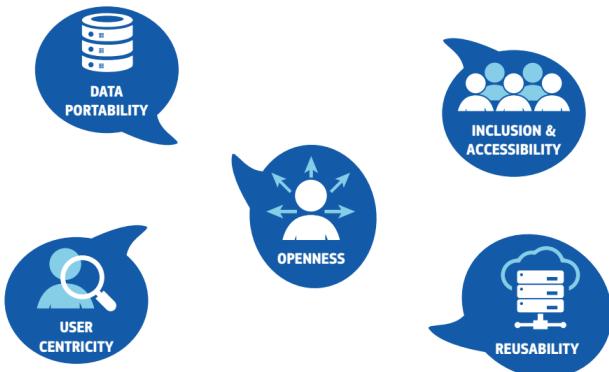
EIF proposes that EU and national legislation and policies are clear, coherent to each other and make good use of digital technologies

ISA² - Interoperability solutions for public administrations, businesses and citizens

https://ec.europa.eu/isa2/eif_en/

EIF INSIGHTS

12 PRINCIPLES driving interoperability actions



47 RECOMMENDATIONS

Concrete and specific as supporting means for implementation



1 MODEL

Designing European Public Administrations

INTEROPERABILITY STORYLINE

FIRST EIF VERSION



SECOND EIF VERSION



THIRD EIF VERSION

Integration vs. Interoperability?

- **Interoperability** has the meaning of coexistence, autonomy, and federated environment
 - “Loosely coupled” means that the components are connected by a communication network and can interact; they can exchange services while continuing locally their own logic of operation
- **Integration** refers more to concepts of coordination, coherence and uniformization
 - “Tightly coupled” indicates that the components are interdependent and cannot be separated
- Two integrated systems are inevitably interoperable, but two interoperable systems are not necessarily integrated

Most referred technology classes

Integrated information systems can be divided into three main classes:

- (1) **Interfaced systems** representing the weakest (but still widely used) form of integration because systems can only exchange data using predefined exchange protocols and data schema (e.g. Comma-Separated Value (CSV) files over FTP (File Transfer Protocol), XML files via TCP/IP and SOAP, SQL schemas over DBLink in the case of Oracle applications, etc.),
- (2) **Tightly-coupled systems** integrating all data sources by creating logical mappings between them using standardised hard-coded interfaces and predefined global schemata and requiring so-called integrating infrastructures such as Enterprise Application Integration (EAI) platforms (Linthicum, 2000) and, in between,
- (3) **Loosely-coupled systems** coordinating autonomous component data sources and software applications with a set of federated schemas and open data exchange formats and protocols, preferably XML formats and using, for instance, Enterprise Service Buses (ESB) for message routing (Chappell, 2004). The latter case equates to interoperable enterprise information systems discussed in the subsequent section. Of course, there could be many intermediate gradations of IS integration between these two extrema (i.e. interfaced and tightly-coupled).

Most referred technology for integration

File Transfer

- Flat files, structured files, XML files

Capturing User Interface – Screen scraping

- Legacy screen scraping
- HTML scraping

Synchronous messages

RPC and RMI Remote Procedures and Remote Methods

- Within local networks – J2EE, .Net

Web Protocols

- Web services for application integration
- REST – for tactical adhoc integration

Data Oriented Integration

- Strongly coupled, development effort involved

API for Package Applications

- Very efficient but requires dedicated know-how

Persistent Messages – MOM

Transaction Oriented – Transactional Monitors

File Transfer

Integration tools have typically a mechanism for transferring and transforming files with various formats

- Flat file, e.g., Comma-separated Values, is an extremely common flat-file format they're easily consumed by Google Spreadsheet, Microsoft Excel, and countless other applications
- Structured files
- XML and JSON files - file transfer universal solving practically all heterogeneity problems
- Stages of communication
 - Encoding – information (object) to file
 - File transfer
 - Decoding – file to object

File Transfer

Advantages

- All operating systems and programming languages support files
- Many applications have ways of exporting or importing files
- Easy to transfer remotely with file transfer protocols, e.g., FTP
- Support disconnected interaction

Disadvantages

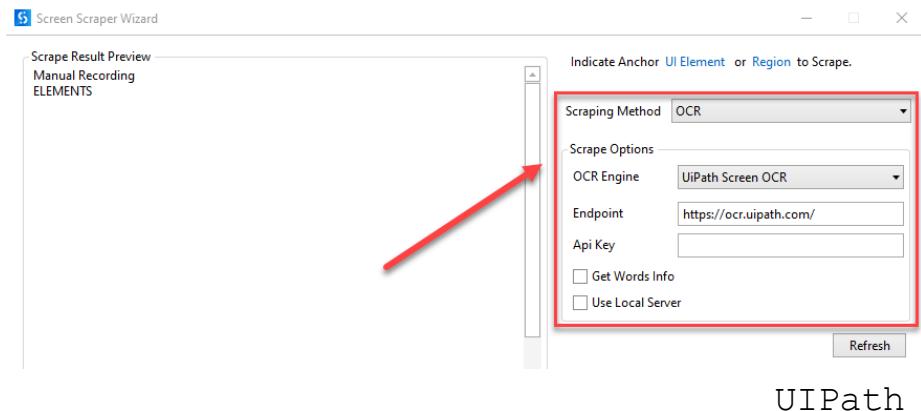
- The complexity of encoding and decoding increases exponentially with complexity of the information to transfer
- Performance is limited

Capturing User Interface – Screen scraping

- Extract information directly from the user interface of an application

- Integration steps

- Define the screens to use
- Create a template indicating input and output fields
- Replace the terminal with a system that simulates a user, sending and receiving data for each screen



UIPath

Web scraping

- Screen scrapers extract information from HTML and other markup languages
- Browsers and other web crawlers use many scraping techniques
- However, most web pages are intended for human consumption and often mix contents with presentation
- Due to widespread scraping, several anti-screen scraping techniques were developed ...



Capturing User Interface – Screen scraping

Advantages

- Suitable for integrating applications with no internal information. For example, COBOL programs on legacy mainframes
- No changes needed in the application
- No direct access to the application data

Disadvantages

- The application interfaces were not designed for integration purposes
- It is not trivial for a program to simulate a user
- The user interface may be volatile
- Performance is low
- Can be unstable due to communication problems, server availability, etc.

Synchronous messages

- In synchronous message passing, the components are processes, and processes communicate in atomic, instantaneous actions called rendezvous. **If two processes are to communicate, and one reaches the point first at which it is ready to communicate, then it stalls until the other process is ready to communicate.** “Atomic” means that the two processes are simultaneously involved in the exchange, and that the exchange is initiated and completed in a single uninterruptable step
- In the synchronous mode problems arise when **communication links or communicating threads are in an erroneous state** (broken links, threads in infinite loops etc.) and thus communicating threads remain blocked, since communication cannot be initiated or completed.

RPC and RMI Remote Procedures and Remote Methods

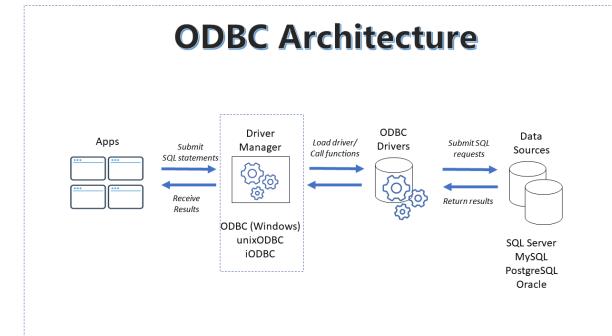
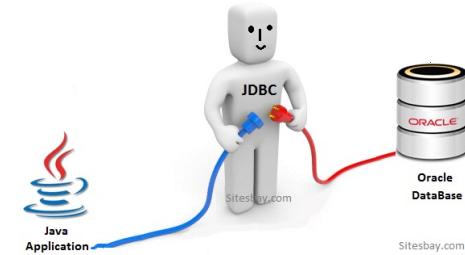
- **Remote Procedure Calls (RPCs)** were developed by Sun Microsystems in the early 1980s. RPCs are represented on different operating systems, including most Unix and MS Windows systems. "Windows NT, for example, supports lightweight RPCs across processes and, with DCOM, full RPCs." (see [15], p.172)
 - **RPC middleware** enables one application **to trigger a procedure** in another application - **running on the same computer or on a different computer or network** - as if both were part of the same application on the same computer.
-
- **Advantages**
 - RPC has a good heterogeneity support, because "RPC has bindings for multiple operating systems and programming languages."
 - Marshalling and unmarshalling are automatically generated, thus simplifying the development
 - **Disadvantages**
 - RPCs don't support group communication
 - They have no direct support for asynchronous communication, replication and load balancing, therefore leading to a limited scalability
 - Fault tolerance is worse than by other middleware types, because "many possible faults have to be caught and dealt with in the program."

Web Protocols

- Many of the released web protocols were used throughout the years to integrate systems. It consists in well-documented connectivity, yet the complexity depend on the details of each protocol
- Web services for application integration, e.g.:
 - **Simple Object Access Protocol** (SOAP) - is a standard protocol originally designed to enable communication between applications developed in different languages and platforms.
 - **Representational State Transfer** (REST) - A stateless client/server protocol: each HTTP message contains all the information needed to understand the request. As a result, neither the client nor the server need to record any state of inter-message communications. In practice, many HTTP-based applications use cookies and other mechanisms to maintain session state (some of these practices, such as URL rewriting, are not allowed by the REST rule). A well-defined set of operations that apply to all information resources: HTTP itself defines a small set of operations, the most important of which are POST, GET, PUT and DELETE. Often these operations are combined with CRUD operations for data persistence, where POST does not exactly fit this scheme.

Data Oriented Integration

- ODBC, JDBC – Independent API for database management systems
- Strongly coupled, development effort always involved
- An application is independent of the DBMS if it doesn't use specific aspects such as stored procedures, triggers, or SQL specific Commands
- Remote processing imposes a performance penalty due to the way parsing and execution commands are made.



Data Oriented Integration

- **Object request broker (ORB) middleware** acts as broker between a request from one application object or component, and the fulfilment of that request by another object or component on the distributed network.
- ORBs operate with the **Common Object Request Broker Architecture (CORBA)**, which enables one software component to make a request of another without knowing where other is hosted, or what its UI looks like - the "brokering" handles this information during the exchange.

Data Oriented Integration

Advantages

- Simple both in Microsoft and Java platforms with ODBC and/or JDBC
- Relatively low cost because it doesn't require rewriting applications
- Most SGDB manufacturers provide drivers

Disadvantages

- A large organization may have hundreds of data bases making it difficult to create an architecture
- The Data Schema must be known
- Requires to have technical knowledge on database repositories because the operation can have serious consequences on the information
- Data types can be different and there is need to transform them
- The data is not validated by the application
- A strong coupled integration – any changes affect the integration
- Replicated data can become inconsistent

API for Package Applications

API (application programming interface) middleware provides tools developers can use to create, expose and manage APIs for their applications so that other developers can connect to them.

Some API middleware includes tools for *monetizing* APIs - enabling other organizations to use them, at cost.

Examples of API middleware include API management platforms, API gateways and API developer portals.

Message-oriented middleware (MOM)

- There are two different types of MOM: message queuing and message passing.
 - Message queuing is defined as indirect communication model, where communication happens via a queue. Message from one program is sent to a specific queue, identified by name. After the message is stored in queue, it will be sent to a receiver.
 - In message passing - a direct communication model - the information is sent to the interested parties. One favour of message passing is publish-subscribe (pub/sub) middleware model. In pub/sub clients have the ability to subscribe to the interested subjects. After subscribing, the client will receive any message corresponding to a subscribed topic
- **MOM** enables application components using different messaging protocols to communicate to exchange messages. In addition to translating - or transforming - messages between applications, MOM manages routing of the messages so they always get to the proper components in the proper order.
- The publish/subscribe MOM works slightly differently. This MOM is an event-driven process. If a client wants to participate, it first joins an information bus. Then depending on its function as the publisher, subscriber, or both, it registers an event listener in the bus. The publisher sends a notice of an event to the bus (on the MOM server). The MOM server then sends out an announcement to the registered subscriber(s) that data is available. When the subscriber requests from a specific publisher some data, the request is wrapped in a message and sent to the bus. The bus then sends an event to the publisher requesting the data

Transaction Oriented or Transactional middleware (TM)

- TM, or even, transaction processing (TP) monitors were designed in order to support distributed synchronous transactions. The main function of a TP monitor is a **coordination of requests between clients and servers that can process these requests**. Request is a "*message that asks the system to execute a transaction.*"
- TM provides services to support the execution of data transactions across a distributed network. The best-known transactional middleware are transaction processing monitors (TPMs), which ensure that transactions proceed from one step to the next - executing the data exchange, adding/changing/deleting data where needed, etc. - through to completion.

What is Middleware *def.*?

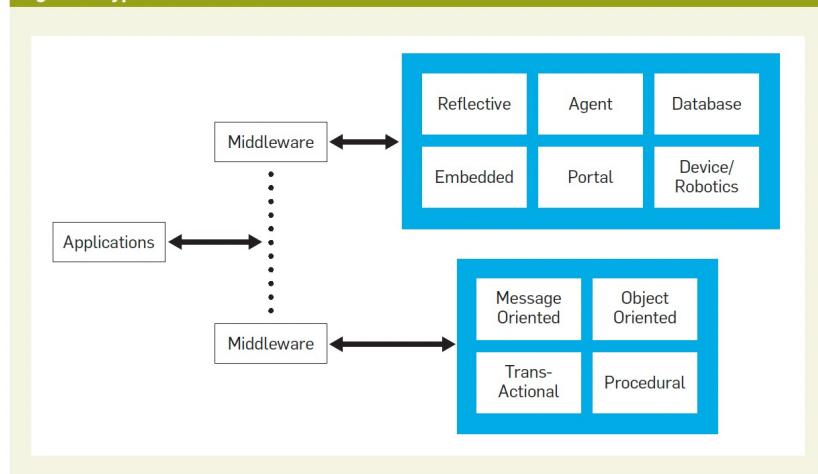
On the one hand, both a software and a DevOps engineer would describe middleware as the layer that “glues” together software by different system components; on the other hand, a network engineer would state that middleware is the *fault-tolerant and error-checking* integration of network connections.

In other words, they would define middleware as **communication management software**.

A data engineer, meanwhile, would view middleware as the technology responsible for **coordinating, triggering, and orchestrating actions** to process and publish data from various sources, harnessing big data and the IoT.

Given that there is no uniform definition of middleware, it is best to adopt a field-specific approach.

Figure 1. Types of middleware.



Middleware – the future

- This middleware “*is an approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.*”
- **This means that middleware should not serve solely as an object-oriented solution to execute simple request-response commands.**

But, middleware can incorporate **pull-push events** and **streams** via multiple **gateways** by combining **microservices** architectures to develop a [holistic decentralized ecosystem](#).

Middleware is STILL Everywhere!!!

Middleware! It was somewhere, and now it is definitely everywhere. Over the past four decades, the term middleware has been tossed around, picked up, and investigated vigorously. **Yet, if you ask 10 different people “what is middleware?” You will most likely get 10 different answers.**

It started as some additions on top of operating systems to **facilitate complex applications development**, moved to become **data integration features**, then became **network applications facilitator**, and eventually became an **important component of every distributed environment, application, system and platform** there is.

To-date, if you examine any type of distributed system or application, you must find middleware or some middleware functionality involved. Although recently, the term itself is becoming less used, yet it still exists in **mobile and sensor networks, service-oriented architectures, grid computing, cloud computing, online multi-player games, networked robotics, the Internet of things**, and much more.

So as is, middleware is really still everywhere and most likely will remain everywhere for a very long time.

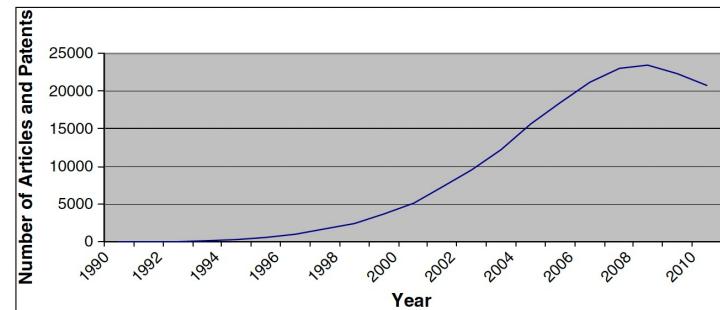


Figure 1. Appearance of “middleware” word on Google Scholar.

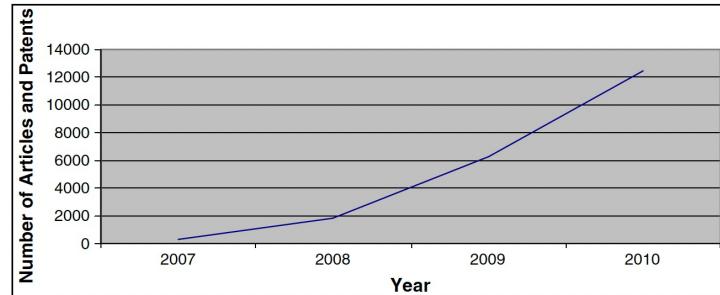


Figure 2. Appearance of “cloud computing” in Google Scholar.

What is Service Oriented Architecture (SOA)?

- The core unit is **the service** that encapsulates a business function, and services are explicitly defined with interfaces that are independent of implementation (**promote the reuse**)
- Services are **loosely coupled** and invoked through communication protocols that are independent of the location
- Each service is instantiated in a **single site** and invoked remotely on this site by all applications that use it (no replicas with potential independent developments)
- There is **no inheritance** or strong dependencies between services
- Each service is created (build) once but can be deployed to all systems that require it.

Results for ALL=“ service oriented architecture” from Web of Science on 8.3.2023

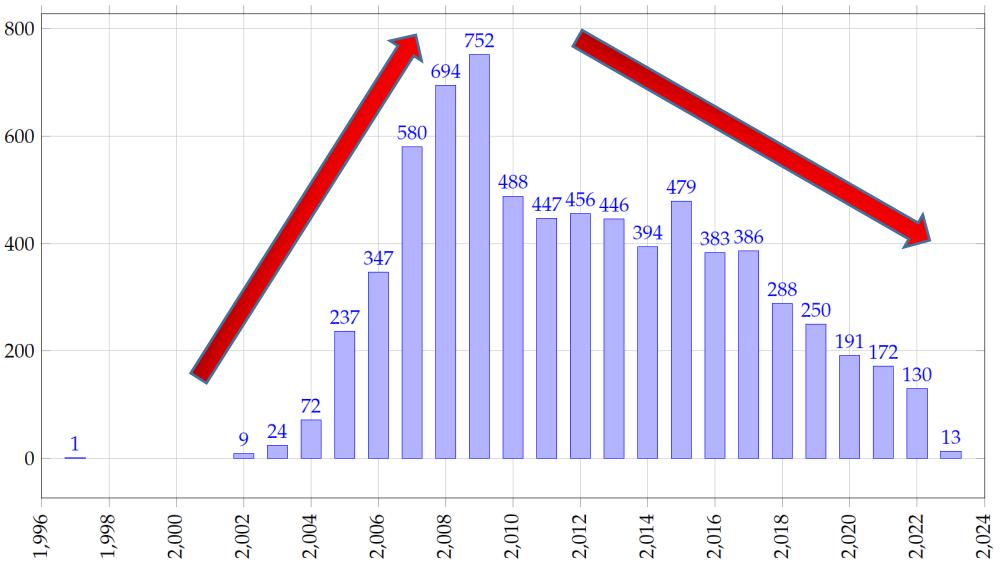
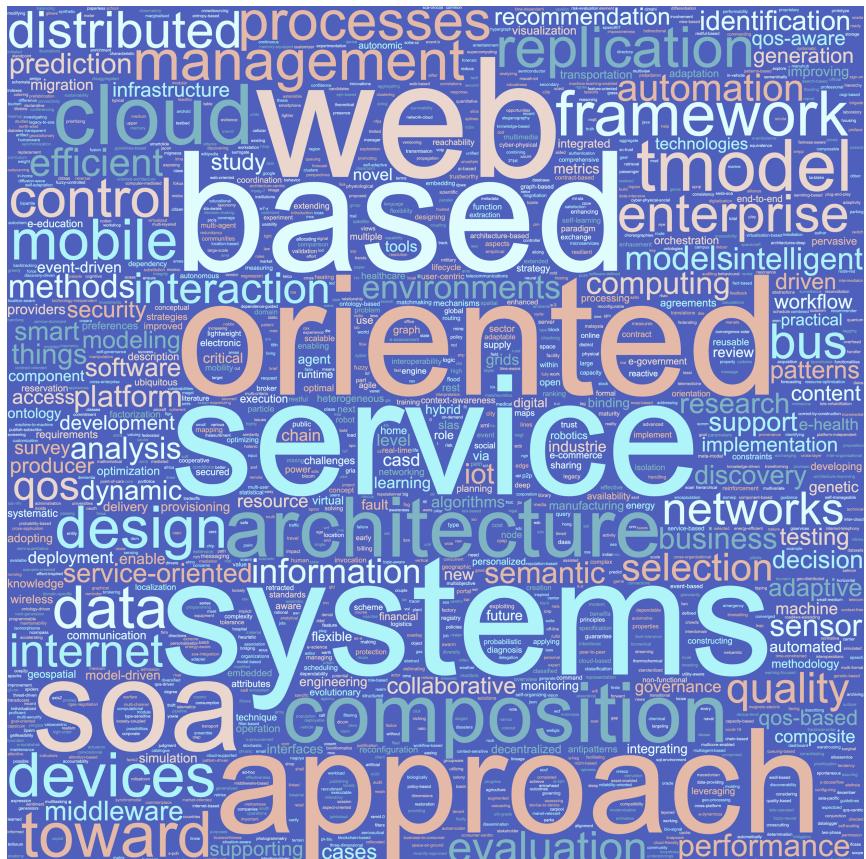
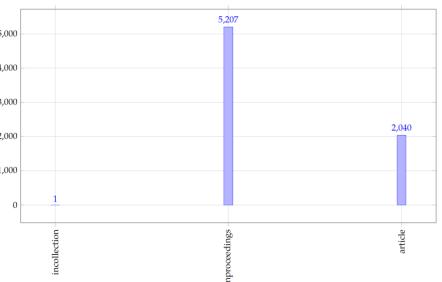
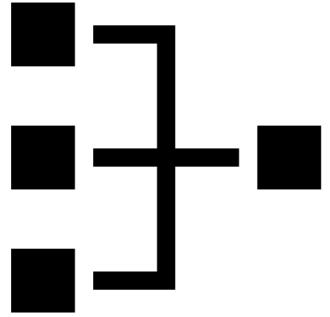


Fig. 3: Distribution of the documents by years. 7248 papers were considered.





Enterprise Service Bus

Available knowledge base and trends

ESB

- An ESB is a system for interoperateing different application systems in a service-oriented architecture (SOA). It represents a distributed computing architecture and is a variant of the client-server model in which an application can assume both a client and a server role.
- The fundamental concept of an ESB is the fact that a communication “bus” is used and placed between the different applications. In this way, applications are decoupled from each other, operating in such a way that they do not need to be aware of the other systems present on the service bus. The complexity of transport services and protocols are abstracted by the service bus, offering applications an easy way to interoperate with each other.
- The ESB concept was born out of the need to escape the point-to-point communication model (applications that interact directly with each other) that proved difficult to manage or monitor when the organization's infrastructure reaches a certain size.
- They also allow accelerating the integration between heterogeneous applications and thus respond to market needs more quickly.
- The mechanisms provided by an ESB platform are the following: connectivity, transactional control, security, data transformation, transmutations, catalog of exposed services, service orchestration, routing.
- **There are two categories for classifying ESBs: (A) open source and (B) commercial ones. Where the criteria of choice usually used to evaluate the ESB are (i) usability, (ii) the support provided, (iii) the functionality provided, (iv) possible adaptability over the functionality provided, (v) the capacity for future expandability , (vi) existing maintenance mechanisms, (vii) community size, (viii) connectivity offered, (ix) cost, and (x) licensing model.**
- **ESB Examples:** Oracle Enterprise Service Bus; IBM WebSphere ESB; TIBCO; Mule ESB; JBOSS ESB; WSO2 ESB

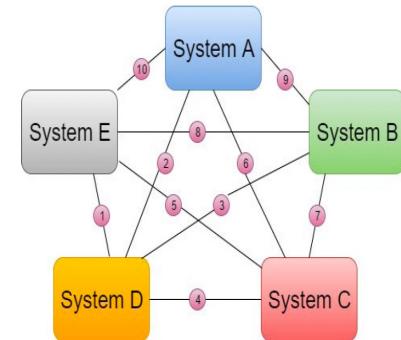


FIGURE 1. Point to point integration architecture.

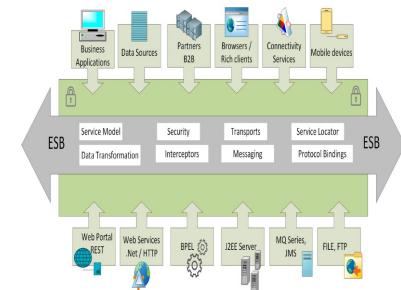


FIGURE 2. Enterprise service bus architecture.

Results for

ALL=“enterprise service bus” AND ALL=ESB

from Web of Science, Google Scholar, IEEE, ACM on 11.1.2023

Technical description

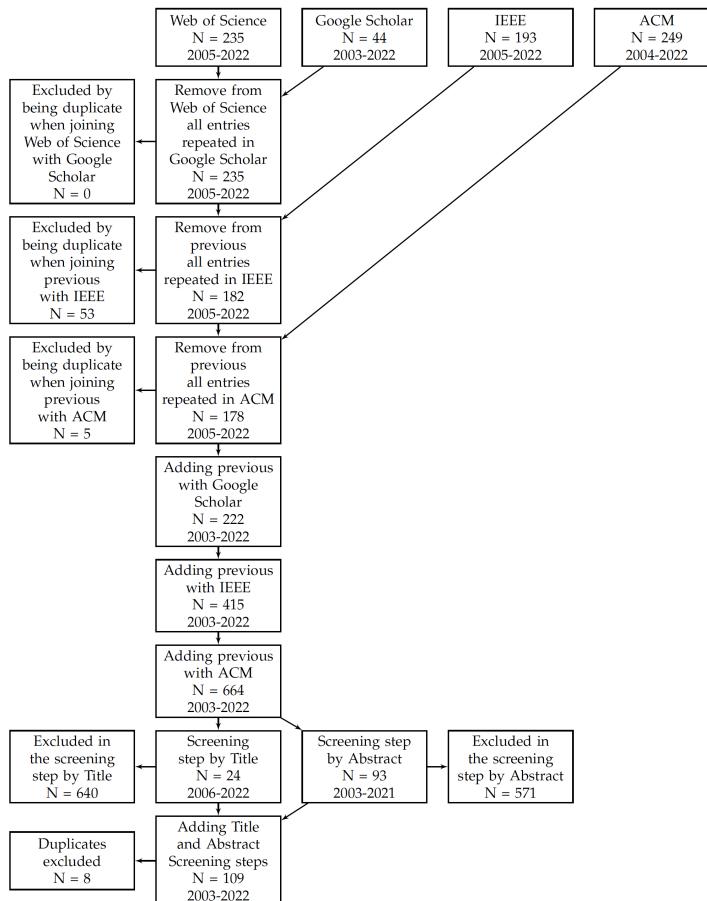


Fig. 1: The refinement steps executed in this SLR procedure and the resulting number of articles.

Results for ALL="enterprise service bus" AND ALL=ESB from Web of Science, Google Scholar, IEEE, ACM on 11.1.2023

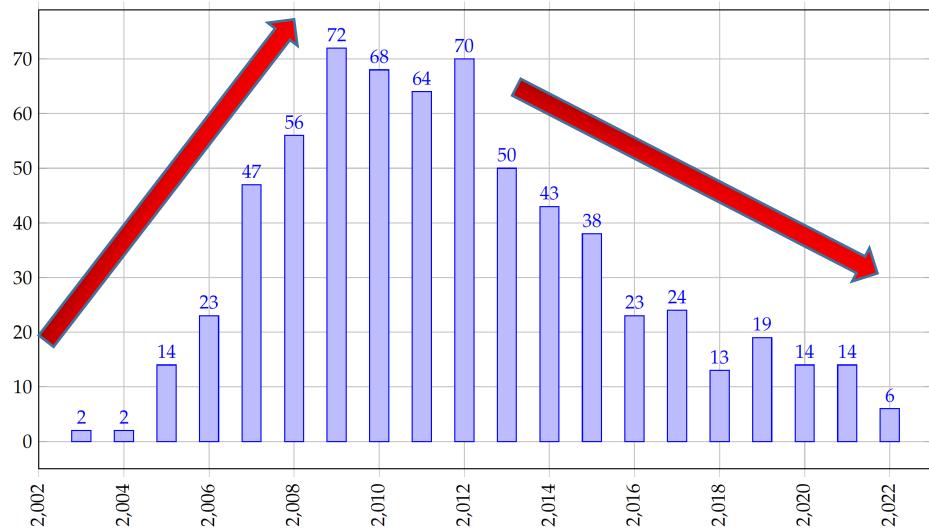
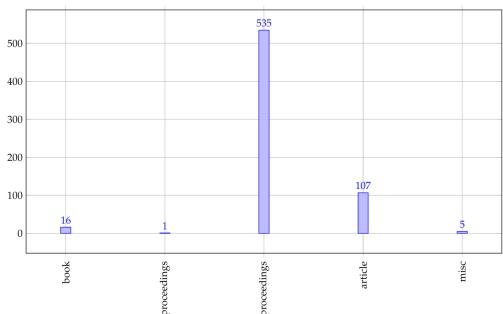


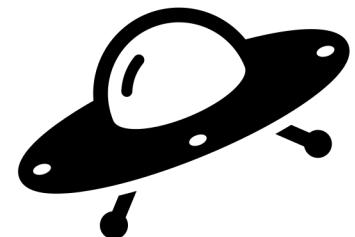
Fig. 4: Distribution of the documents by years. 664 papers were considered.



Messaging-Based Networks reuse STG empirical Fault Provisioning via Design der Runtime des Challenges OPERANDO Life-cycle Integrated Techniques SOA-Integration ... Discover Ein monitoring Apache Two Technology-Independent Exchange How 1 2 Experience Flexible 3 Technical Analysis Part Oriented Praxis Clocks Building role COMPONENT-ORIENTED Theory Selection fundamental utility GraphQL? ProBus Open Comparison Enable Based Protocol Gaps Jordan Process-Based latest Communication Sign-On Context Analyzing Composite one service-based Liang-Jie ESB-Schrift Spezifikation entrepreneur Heterogeneous meet Choose HARDWARE Distributed Your Microsoft's Event Paradigma Pengembangan Comprehensive Architectures through bus epitome Plenary Rawat Comparative (EAII) Test zur Rancang Executable BUSINESS Jalan Balancing Interaction Structured Anatomy Terintegrasi Algorithm laboratory Problem Static Geschäftsprozess-Automatisierungen Adapt reasons Scalable Industrial applied Directions concept Corporate Schritt-Einsatz Self-Managing Evaluating Review Modeling Way Processing buzzword Wide Robustness Bangun Component Spoilt middleware over Composition interfaces Coordination JBoss and Instalasi (SOA)Concepts Grundlage XaaS departmental Within Metric developer target-transparent injection Handling choreographies illustrations Lecture Platform Support Lasting Security Gossip-Based Aspect Shared Processes signatures executive Aplikasi real-time service-oriented Goldschlager Organisations centers Geschäftsmodelle Vector dashboard Theoretical Load Exploring dependency Collaborative Definition (ESB) Queuing IT Cloud-Based Cycle Survey system Graph solutions"Services Web among Designing Life Resources Contract Application MAS Autonomic server integrating Advanced underpinnings iPortalDoc-Enterprise Pervasive enterprise consolidating methods Composable Right REPLICATED Research Computing Flexibility Profile Implementation Mechanism Case Complex government Bridging ESB low Event-Driven ESB-Enterprise power mediation Reliable basierender Study Pattern SOA Tools Routing PublishSubscribe Educational Electronic-Commerce Unified Mobile Smart Holistic Conceptual help Telecom framework When UDDI evolving Strategies Model Implementing Suites-Make for orchestration Choice HIGH Paradigm data ServiceMix Semantic Template Technology Practical Tenets new DIFFERENT Virtualizing MS adapting als Why Handler Decentralized towards Resilience ESB-Based Necessary Self-Organization Realization CA-Graph Functionality Enhancing Approach eines Environment Collection Software MVC Notifications dengan Steps behind Dynamic Scalability Re-Configuration GEMBus Bus-Grundlagen Konzepte Information Licensing Source Optimization Interoperability Menggunakan Engineering Context-Aware Monitor Solutions QoS UML für (BMP) consumer requirements INTERNET Zhang—Designing electric abstraction Supporting control WS-Monitor Management Improving WS-Policy-Aware intelligent early-warning Aware Single adaptations XML Cloud CONSTRUCTION Strategy Future MODEL-BASED Perfomance Service

Fig. 2: Wordcloud of the main concepts extracted from Title.

The future of Enterprise Integration



Event-driven Systems

Available knowledge base and trends

Results for

("event-driven" OR "event driven") AND (system OR systems) AND (integration OR middleware OR EAI OR ESB OR "enterprise application integration" OR "enterprise service bus")

**from Web of Science,
Google Scholar, IEEE, ACM
on 11.1.2023**

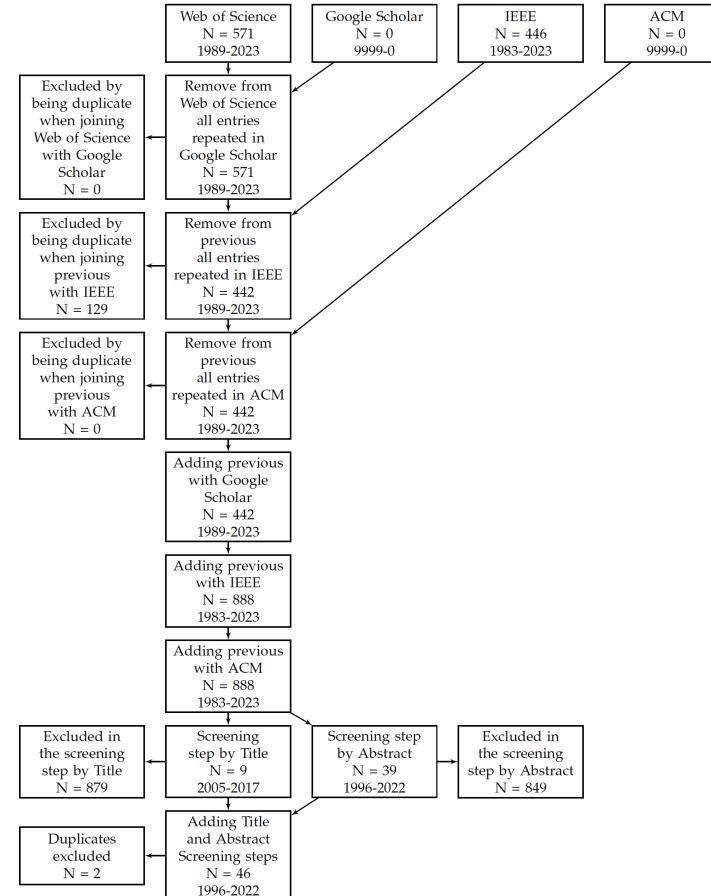


Fig. 1: The refinement steps executed in this SLR procedure and the resulting number of articles.

Technical description

Results for ("event-driven" OR "event driven") AND (system OR systems) AND (integration OR middleware OR EAI OR ESB OR "enterprise application integration" OR "enterprise service bus") from Web of Science, Google Scholar, IEEE, ACM on 11.1.2023

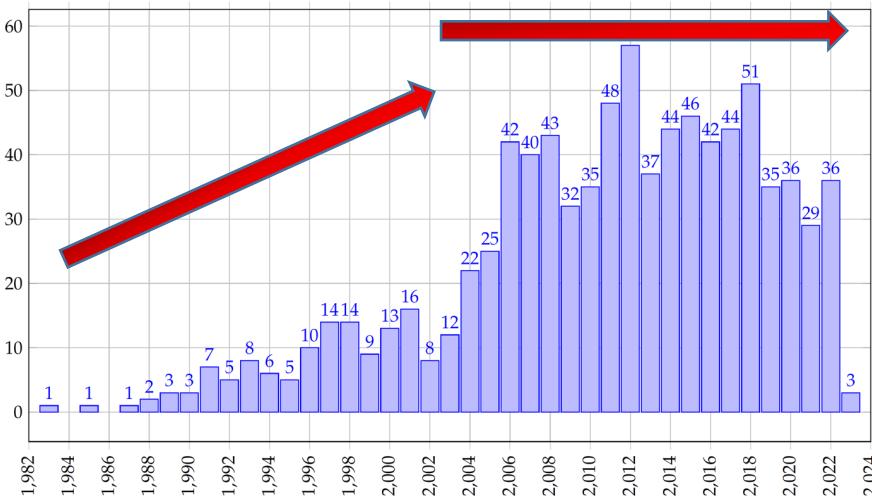
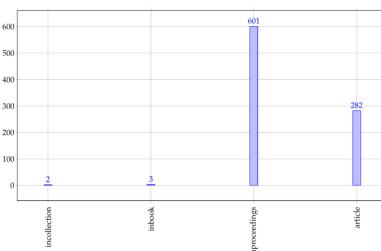


Fig. 4: Distribution of the documents by years. 888 papers were considered.



architectures Trust-based Server discrete-event Towards grids supply Elaborating robot complex manufacturing Foundation knowledge energy based Web-enabled Things impact reengineering Workflow active decentralized monitoring analysis easy planning Flexible Technical multicenter Effect placement system-on-chip object Building sophisticated forest event-based Big-Data operating acquisition WoT fluxes business decision GLEE MAS4AMR rescheduling IoT@Work CH4 marketing collaboration information plasma open bus agent medical rules express robotic Industry multi-segment dynamic Event-driven image dispatching test develop traceability modelling fusion final exchange Internet support next neural Lightweight production representation Processing mixture supervision BESA-ME reporting middleware enhancing effectiveness coordinating interfaces mobile technology message environment management micro 4.0 Secure CSCW configurable practice software movie simulation interactive approach tracking process-oriented Piezoelectric multi computer-based compatible real-time experiment safety Development centers guidelines Contiki model case generation method large electrical classification PVDF intelligence system integration Designing pervasive Current study periodic pattern covariance practice-driven distribution smart modern low TAXII Events model-driven power Trends fault-tolerant ledger chain SOA SOC FJM2 health Verification lots JMS framework unified organized self cyber-physical measuring auto data ALIAS autonomous experience rule-based trial automation solution communication XMPP observability traffic new LSI GIS-based DSS DEVICE runtime film Resilience clinical context-aware sensor automotive chicken Scalability scientific scale Windows platform integrated switch Earth asynchronous design physical High-performance fuzzy web-based HARPS3 execution process cost supervisory implementation resources control human-robot Opportunities cyber application threat airbag multiagent methodology

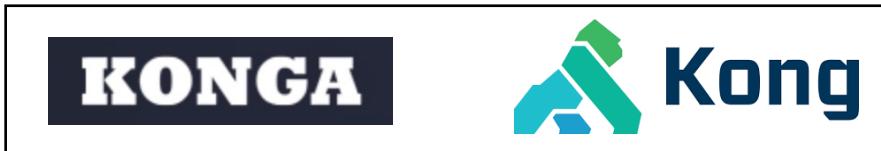
Fig. 2: Wordcloud of the main concepts extracted from Title.

The technology that you'll be using is event-driven focused

Business
Processes
execution



API
Management



MicroServices

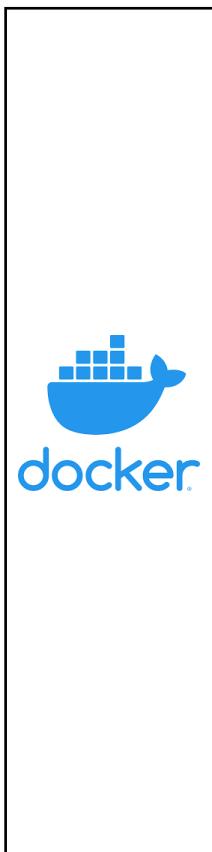


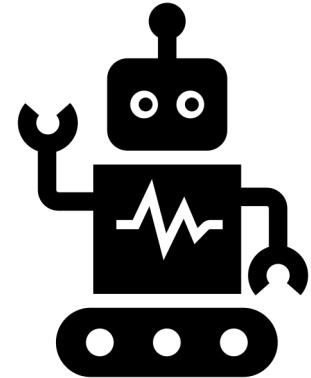
Messaging



PaaS
IaaS

Hyperautomation





Hyperautomation

Available knowledge base and trends

Results for

**hyperautomation (All Fields)
OR hyper-automation (All
Fields) OR "hyper
automation")**

**from Web of Science,
Google Scholar, IEEE, ACM
on 16.1.2023**

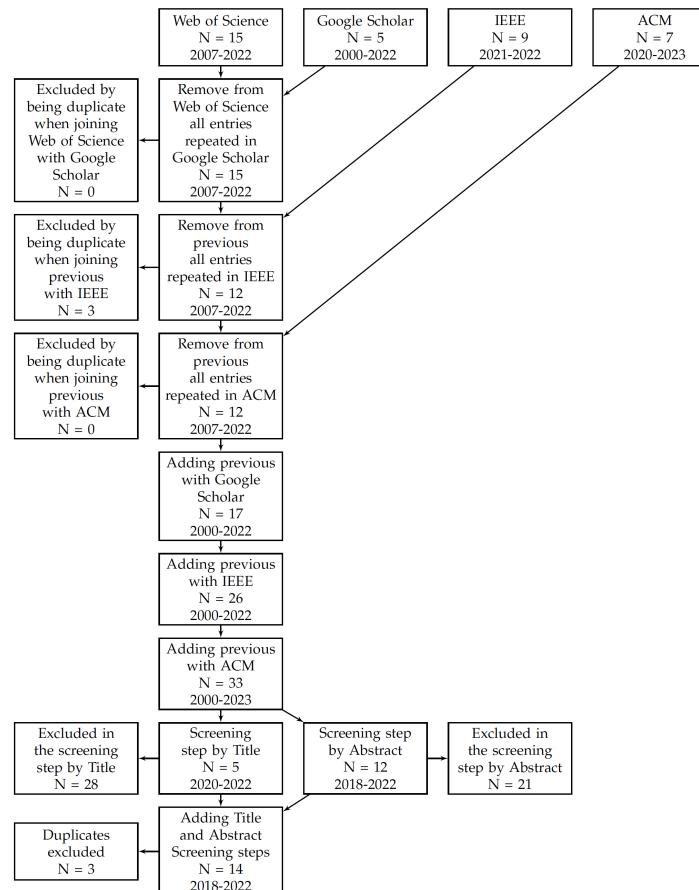


Fig. 1: The refinement steps executed in this SLR procedure and the resulting number of articles.

Technical description

Results for hyperautomation (All Fields) OR hyper-automation (All Fields) OR "hyper automation") from Web of Science, Google Scholar, IEEE, ACM on 16.1.2023

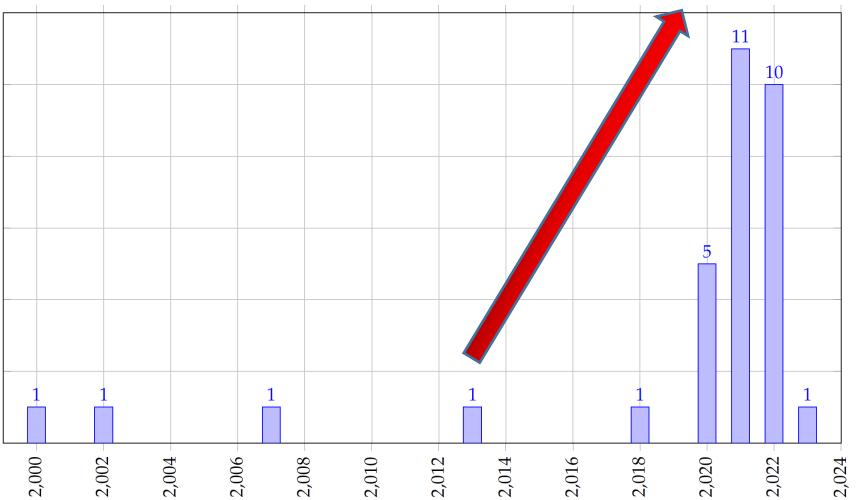


Fig. 4: Distribution of the documents by years. 33 papers were considered.

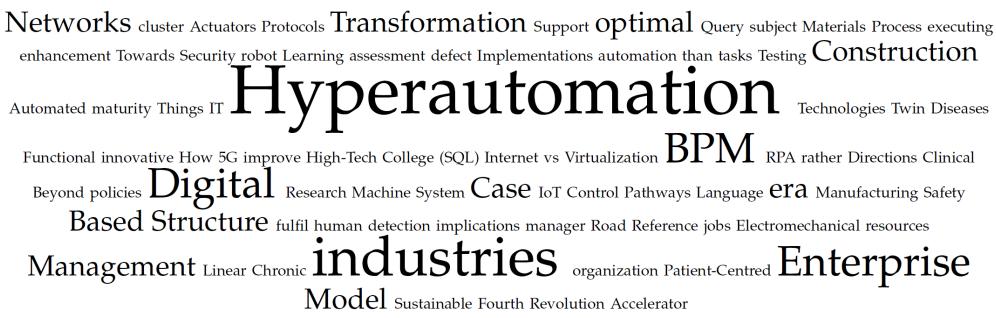
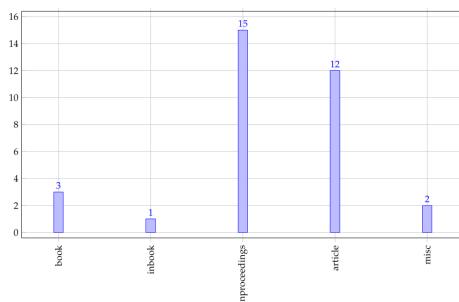


Fig. 2: Wordcloud of the main concepts extracted from Title.

Example of new IE applications: blockchain interoperability

A. The need

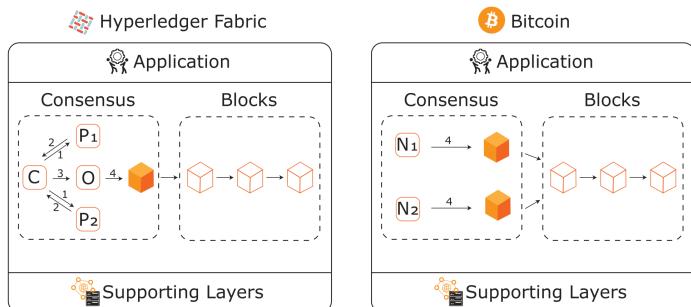


Fig. 2. Representation of two blockchains, Hyperledger Fabric [7], and Bitcoin [132].

C. The research investment

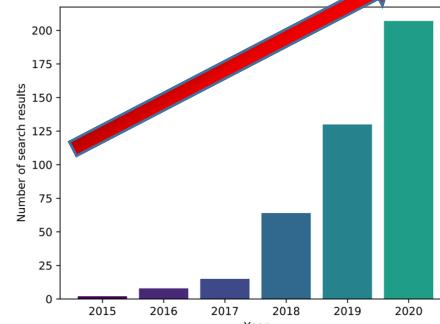


Fig. 1. Research trends on blockchain interoperability.

B. The conceptual solution

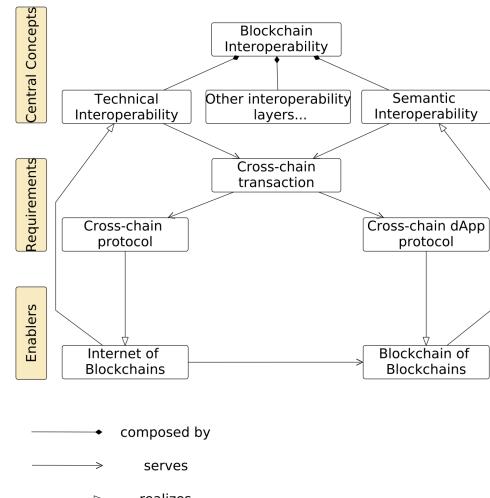


Fig. 3. Concept map, illustrating the relationship between different concepts related to blockchain interoperability

Contemporary integration strategies used for blockchain interoperability

Table 1. Survey Comparison Criteria, Description, and Sub-Criteria

Criteria	Description	Sub-criteria 1	Sub-criteria 2	Sub-criteria 3
Public Connectors (PCs)	Addresses public connectors	Sidechains	Hash lock contracts	Notary Schemes
Blockchain of Blockchains (BoBs)	Addresses BoBs	Describes solutions	Detailed comparison	N/A
Hybrid Connectors (HCs)	Addresses Hybrid Connectors	Trusted Relays	Blockchain agnostic protocols	Blockchain migrators
Architecture (AR)	Addresses architectures enabling CCCPs	Proposes architecture	Presents related work	N/A
Cross-chain Standards (ST)	Addresses standards for interoperability	Present standards	Relate standards to solutions	N/A
Cross-analysis (CC)	Compares across categories	Compare categories	Compare sub-categories	N/A
Use Cases (UCs)	Presents use cases using an IoB or BoB	Existing use cases	Predicted use cases	N/A
Open Issues (OIs)	Challenges on interoperability	Research directions	Relate interoperability to other issues	N/A

Q&A





TÉCNICO LISBOA

Enterprise Integration

Messaging systems

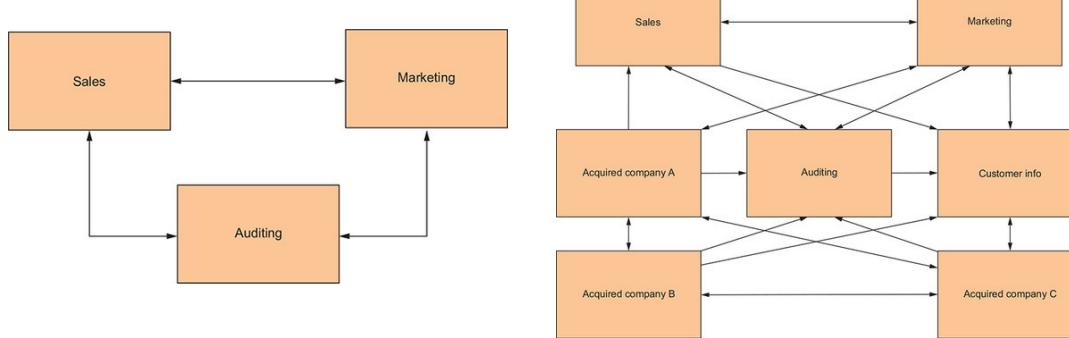
Prof. Sérgio Guerreiro

Sergio.guerreiro@tecnico.ulisboa.pt

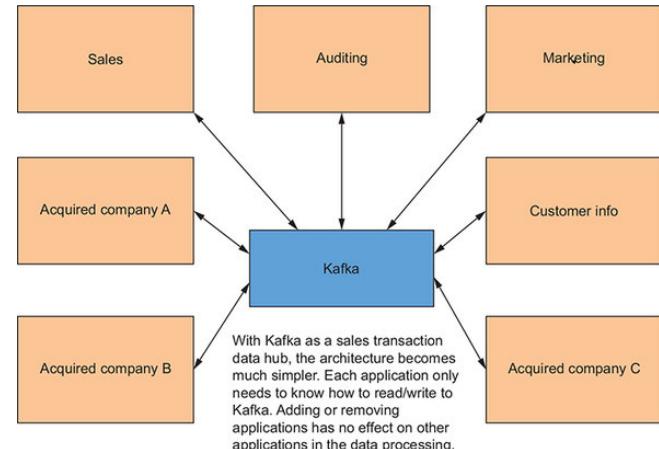
Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

The problem, then Publish and Subscribe solution



- The original data architecture for ZMart was simple enough to have information flowing to and from each source of information.
- With more applications being added over time, connecting all these information sources has become complex.
- Using Kafka as a sales transaction hub simplifies the ZMart data architecture significantly. Now each machine doesn't need to know about every other source of information. All they need to know is how to read from and write to Kafka.



Publish/Subscribe Messaging

- Publish/subscribe (**pub/sub**) messaging is a pattern that is characterized by the sender (**publisher**) of a piece of data (**message**) not specifically directing it to a receiver. Instead, the publisher classifies the message somehow, and that receiver (**subscriber**) subscribes to receive certain classes of messages.
- Pub/sub systems often have a broker, a central point where messages are published, to facilitate this pattern.

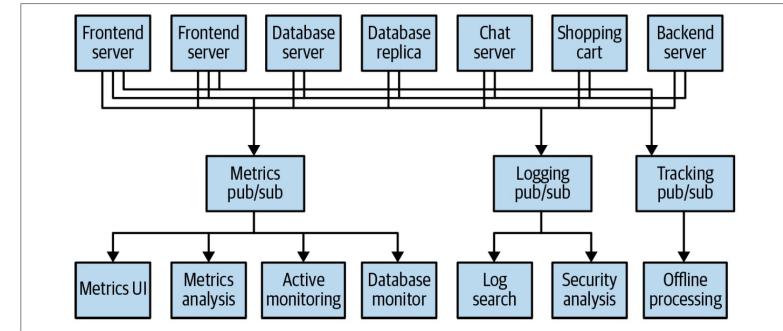


Figure 1-4. Multiple publish/subscribe systems

Limitations of the synchronous communication

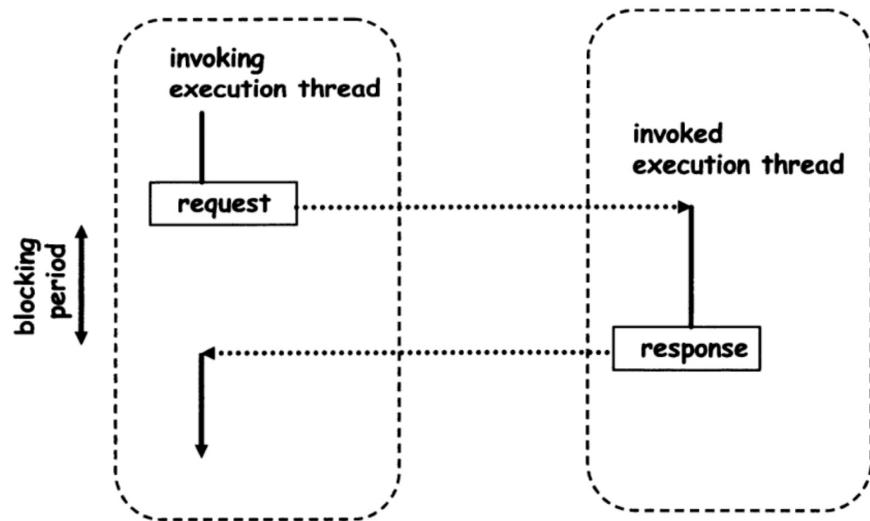


Fig. 1.14. A synchronous call requires the requester to block until the response arrives

- Both applications must be running simultaneously – time coupling
- Blocking period exist
- Fault on communication
- Fault on threads
- Different receiver/production capacity rate => message lose

Asynchronous communication

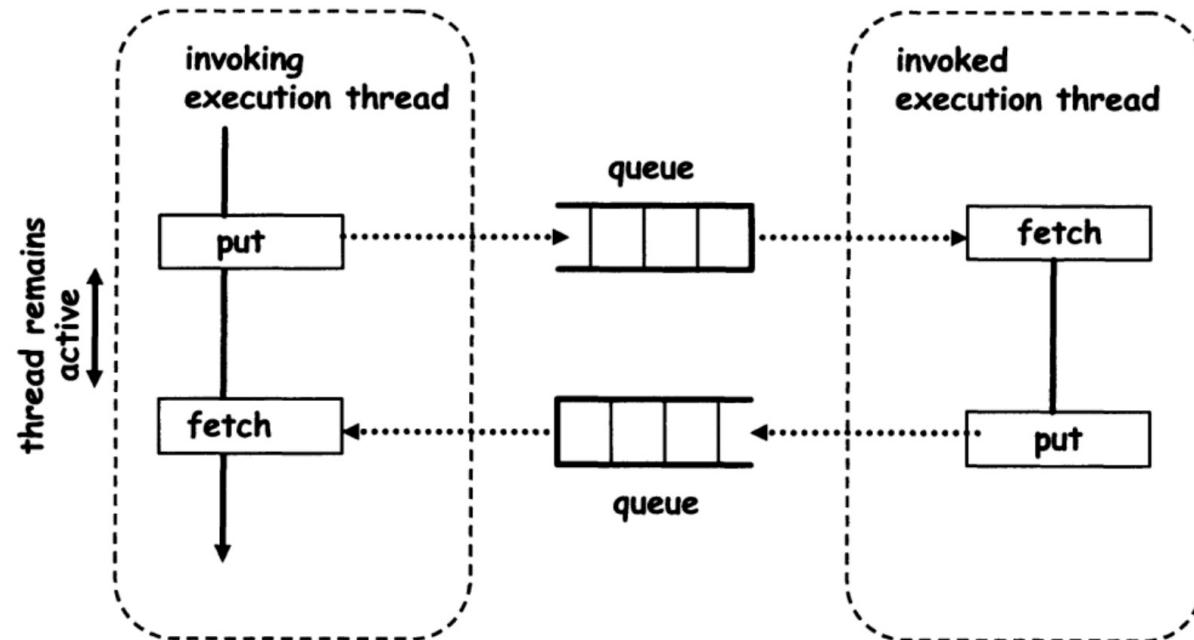
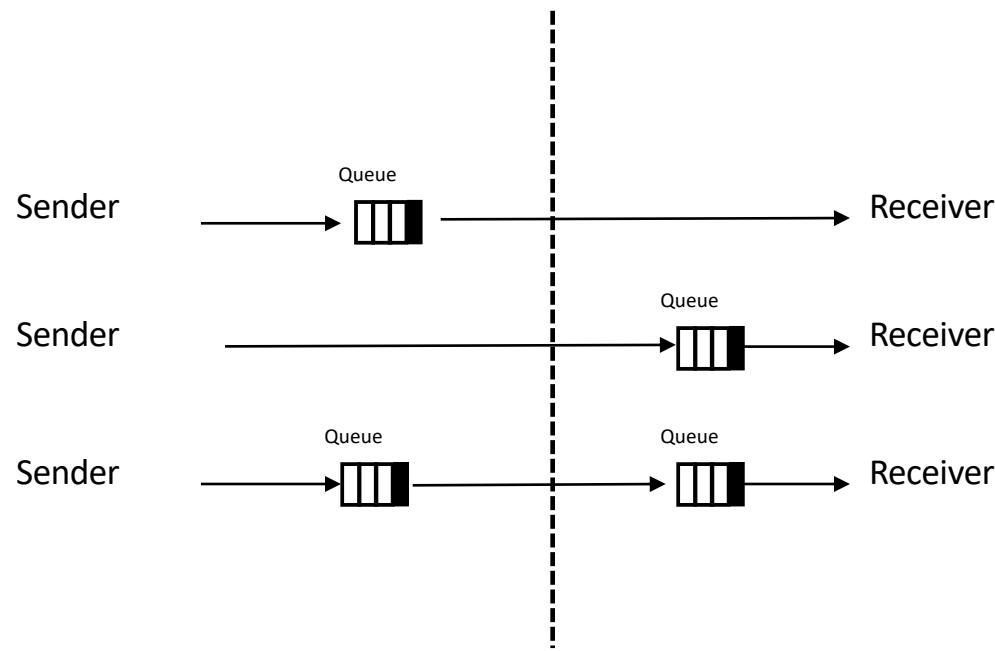


Fig. 1.15. An asynchronous call performed through queues allows the caller to continue working while the request is processed

Solutions for Asynchronous communication with messages queues

Other communication models designs are possible:



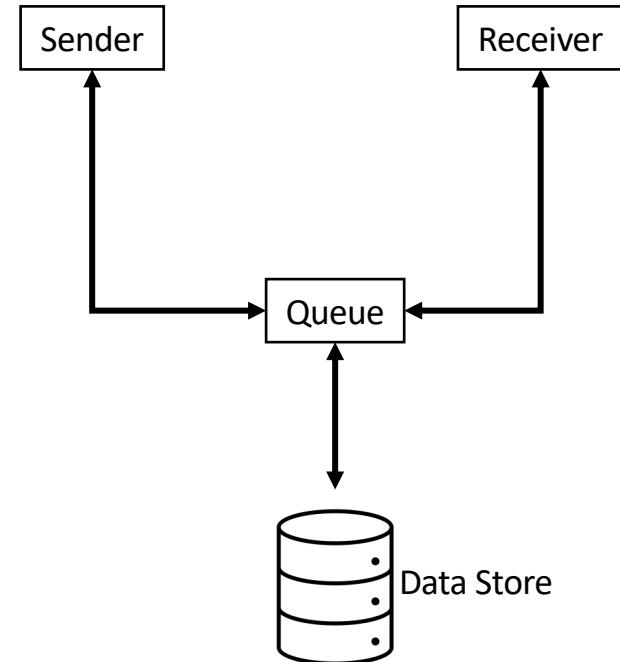
Time decoupling vs. asynchronous

- Time decoupling and asynchronous communications are different concepts
- Time coupling implies that both the sender and receiver of the information on an integration must be running simultaneously, or the communication fails.
 - Where, the service invocation may be synchronous or asynchronous
- In time decoupling there is no requirement that both are up and running, tolerating transient crash faults of the server
- **Obviously in time decoupling only an asynchronous communication pattern makes sense. Therefore, to have time decoupling, we need some form to persist messages**

Message Oriented Middleware (MOM)

Advantages

- The capability of persistently storing messages (Store forward):
 - Buffering messages decouples the rate of production and the rate of consumption, unblocking the sender
- Allows to cope with the unavailability of the Service.
- Tolerates temporary crash faults of the service
- Act as a broker to distribute messages accordingly to different routing patterns



Disadvantages

- Asynchronous communication implies a less intuitive programming model (like event programming) than the request-response paradigm
- Message queues need to be supported and managed with additional support and investment costs

MOM key concepts

Channel

- Corresponds to message queues with unique name.
- Guarantees persistence.
- Point-to-point – dedicated channel to deliver unique message
- pub/sub – a copy of a message for each client

MOM key concepts

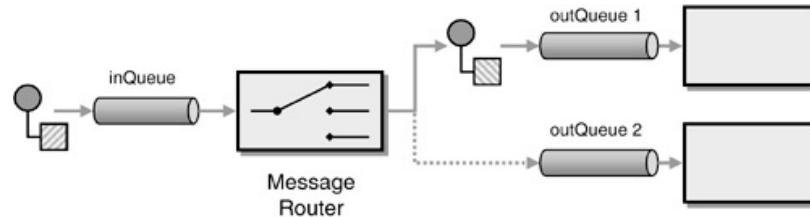
Message

- Package of data that is passed through a channel
- Could be: document, event, image, video, etc...
- To allow interoperability the same schema is required, otherwise message need to be converted

MOM key concepts

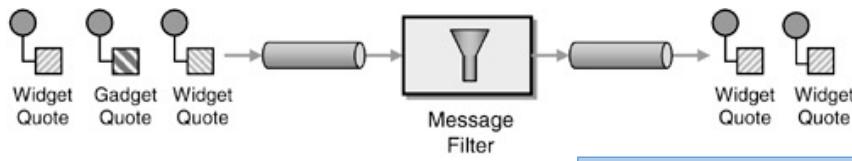
Router

- Is the component in the messaging system that decides the destination of a message
- Different messages may be handled by different applications
- The routing decision is handled in the messaging system

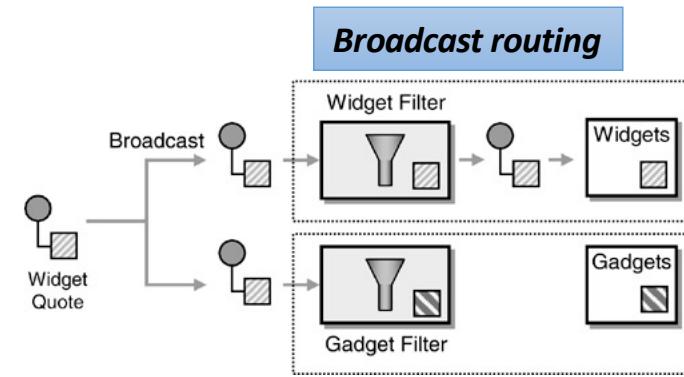


MOM key concepts

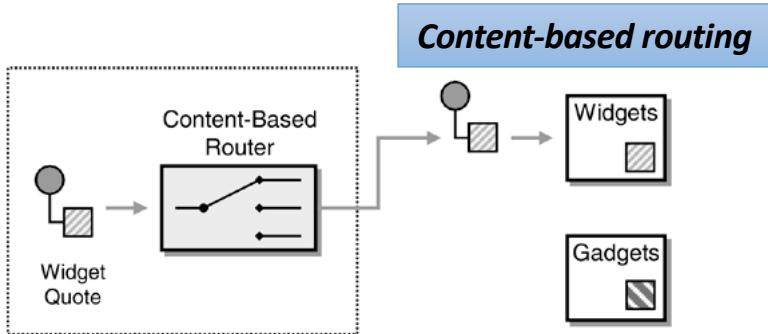
Other **routing** mechanisms



Message-type filter

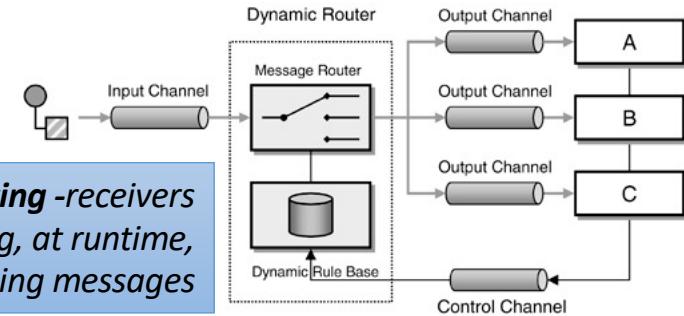


Broadcast routing



Content-based routing

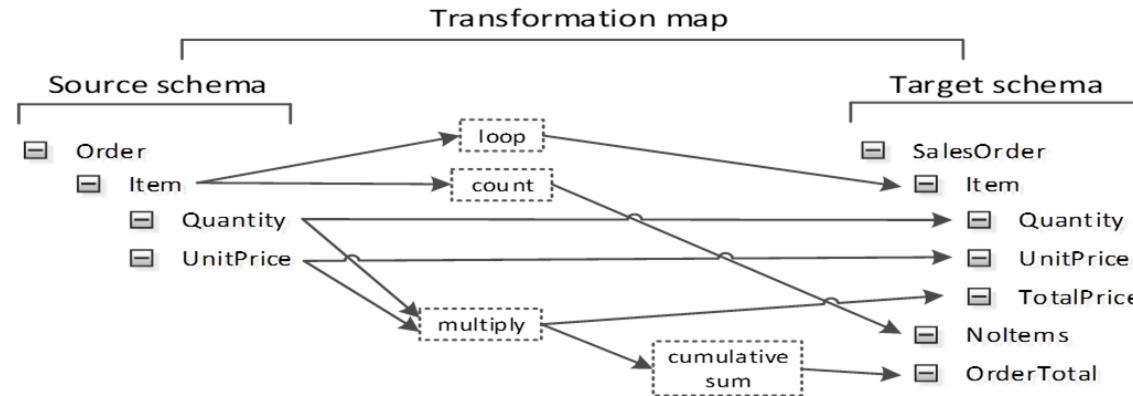
Dynamic routing -receivers configuring, at runtime, rules for routing messages



MOM key concepts

Translator provides the ability to convert message content from one structure into another

A transformation map defines the translations required:



MOM key concepts

Endpoint is the software component that abstracts the sender and receiver application from the messaging system.

For sending messages: invoke the send method with the desired mode (e.g., fire-and-forget, synchronous or asynchronous)

For receiving messages:

Pooling: receiver checking in the messaging system if there are any new message available

Callback: receiver is notified by the messaging system when any new message is available

Advanced Message Queuing Protocol (AMQP)

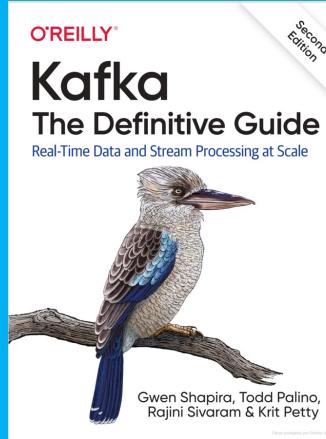
Historically, there was a lack of standards governing the use of message-oriented middleware. Most of the major vendors have their own implementations, each with its own application programming interface (API) and management tools.

The Advanced Message Queuing Protocol (AMQP) is an approved OASIS and ISO standard that defines the protocol and formats used between participating application components,
<https://www.amqp.org/>

AMQP specifies flexible routing schemes, including point-to-point, fan-out, publish/subscribe, and request-response, transaction management, queuing, distribution, security, management, clustering, federation and heterogeneous multi-platform support.

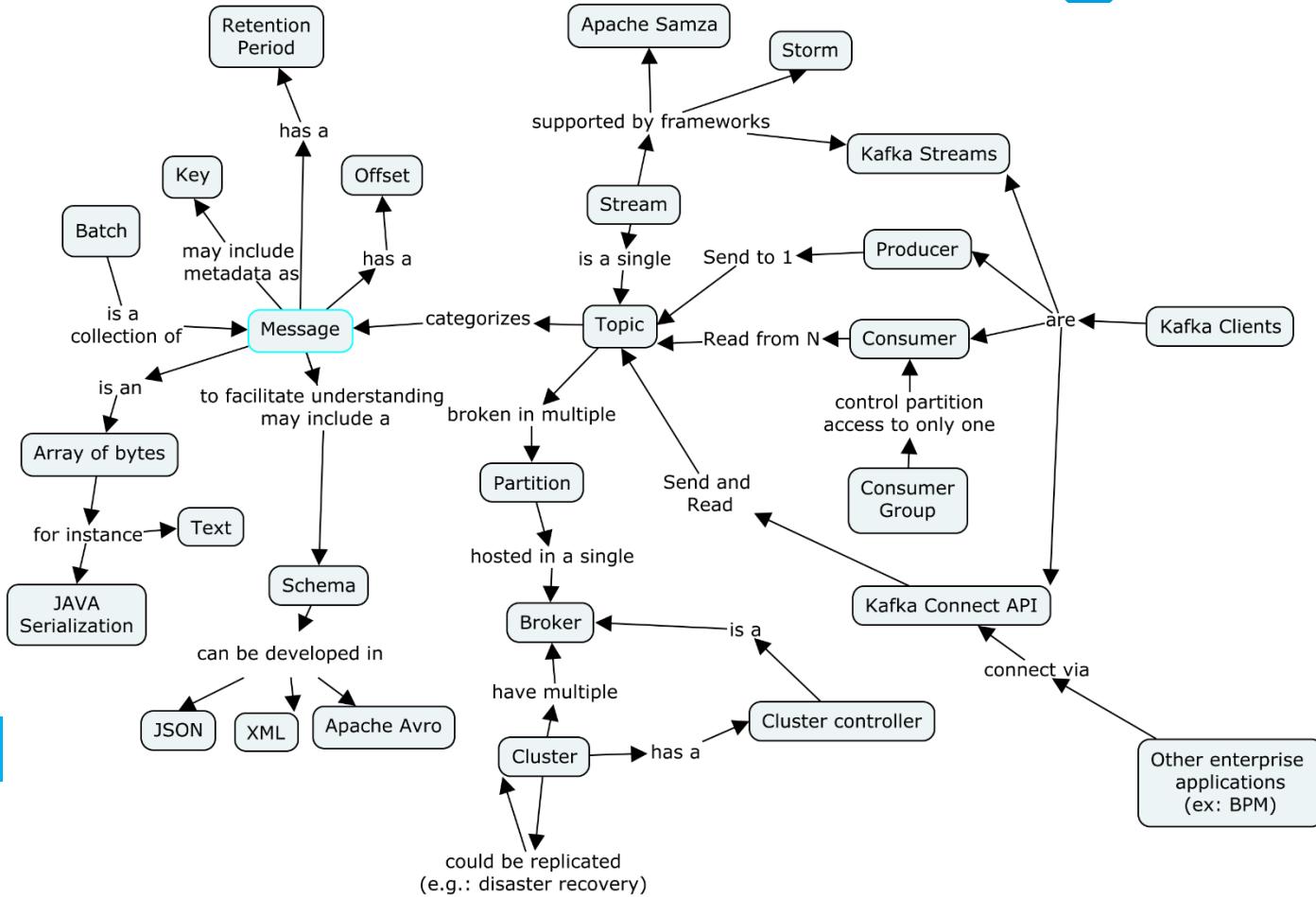
Example of AMQP implementation: <https://www.rabbitmq.com/>

Kafka



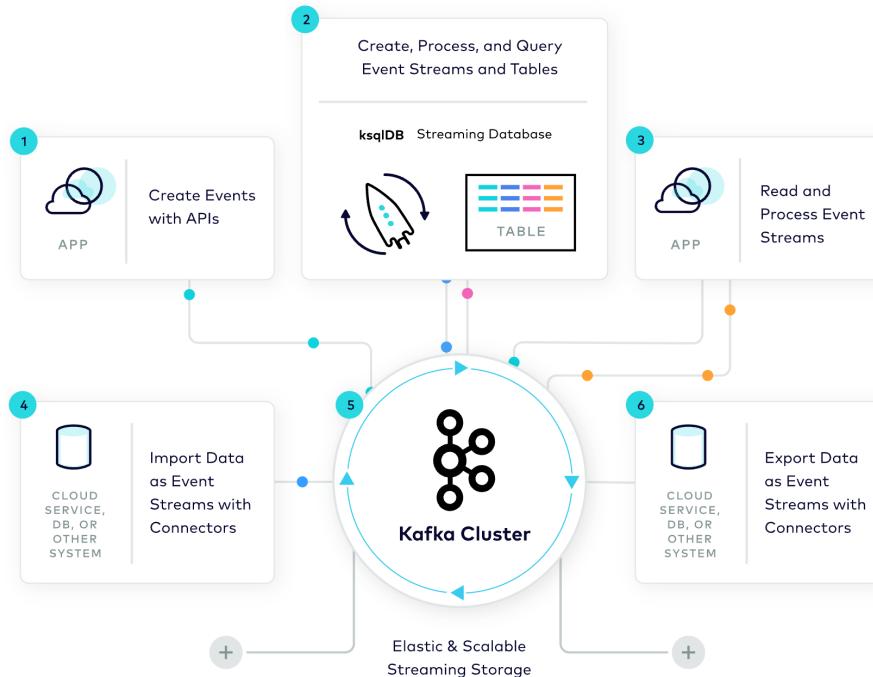
Shapira, G., Palino, T., Sivaram, R., & Petty, K. (2021). Kafka: the definitive guide. " O'Reilly Media, Inc."

Kafka Conceptual Map



What Does Kafka Do?

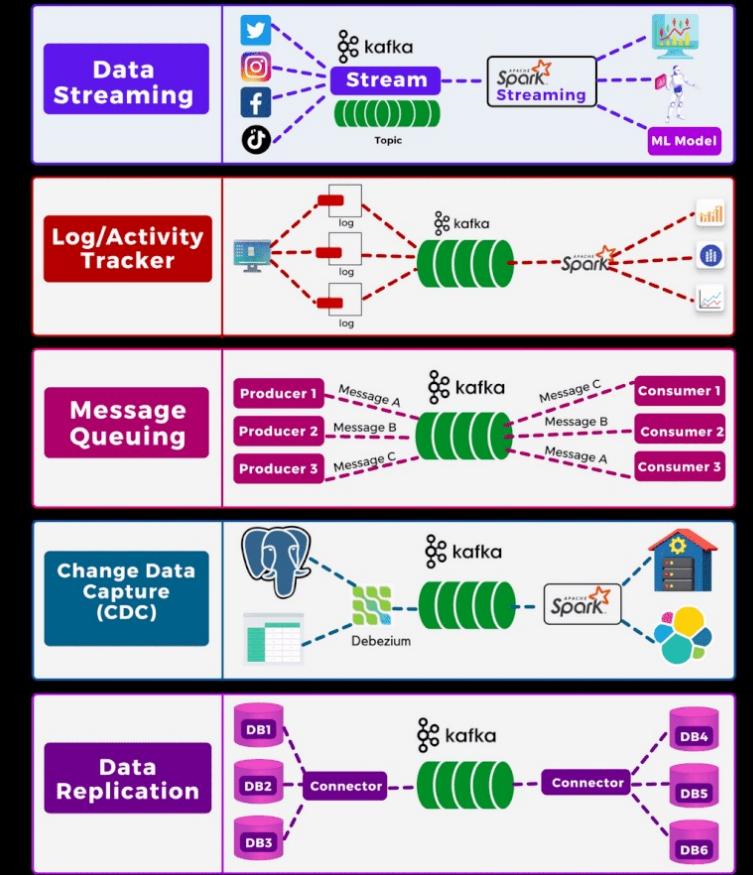
Learn about the fundamentals of Kafka, event streaming, and the surrounding ecosystem. Click on an element to find out more.



Brij Kishore Pandey

DON'T FORGET TO SAVE

TOP 5 KAFKA USE CASES 2.0

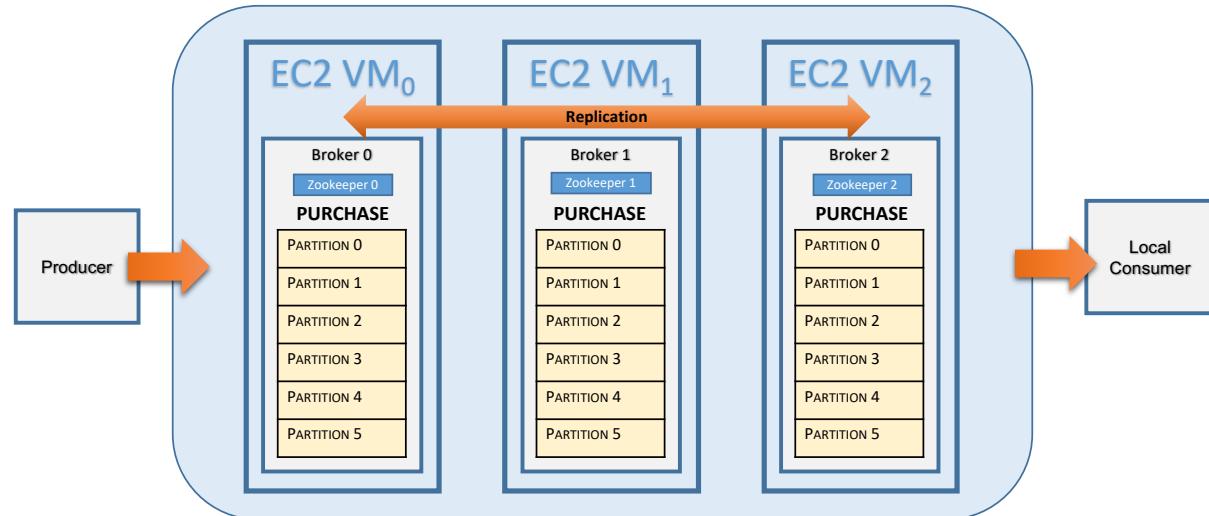


Kafka Architecture



Kafka and Zookeeper

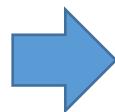
- ZooKeeper is reliable storage running on a cluster of nodes
- Kafka uses Apache ZooKeeper to maintain the list of the brokers and all the information related to partitions, offsets, *etc.*
- ZooKeeper can guarantee that the local replicas never diverge



MOM versus Kafka

- MOMs provide:

- Reliable communication in presence of transient faults on consumers
- Buffering between producers and consumers
- Publish/subscription and point to point channels



- Kafka adds with:

- High ingestion rate of messages
- Distributes architecture tolerating faults from brokers
- Disk retention policies

- MOMs normally provide

- Message transformation
- Transactional messages
- Automatic indexes for reading
- Dynamic routing of messages

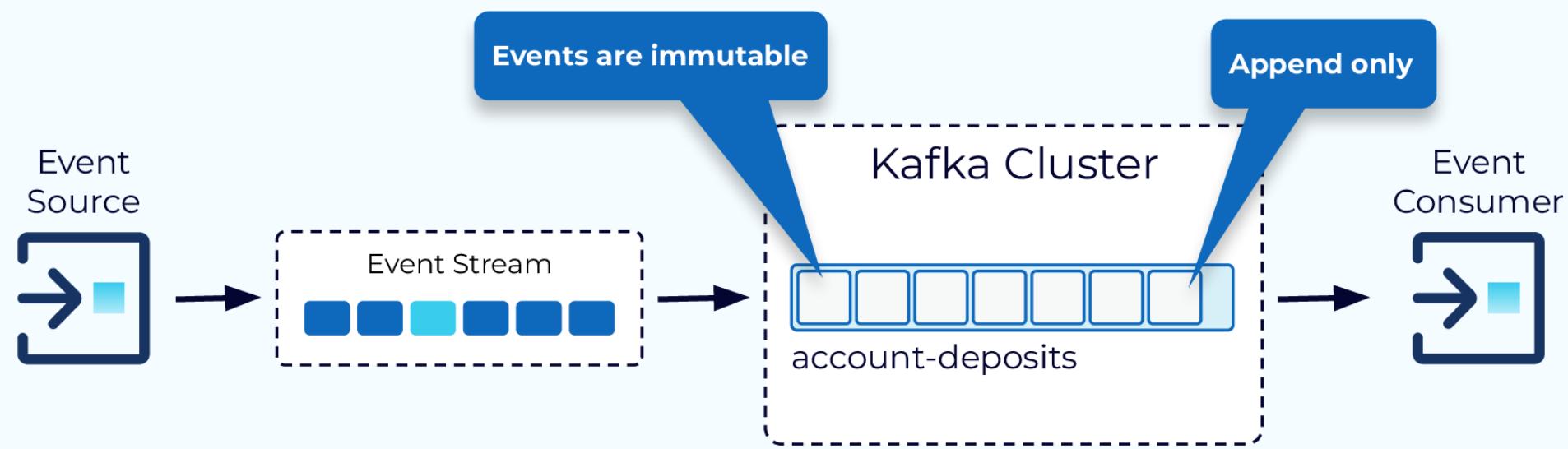


- Kafka does not provide any of these features

Commit log

- ...a commit log is designed to provide a **durable record of all transactions** so that they can be replayed to consistently build the state of a system
- Similarly, data within Kafka is stored durably, **in order**, and can be **read deterministically**
- In addition, the data can be **distributed** within the system to provide **additional protections against failures**, as well as significant opportunities for **scaling performance**

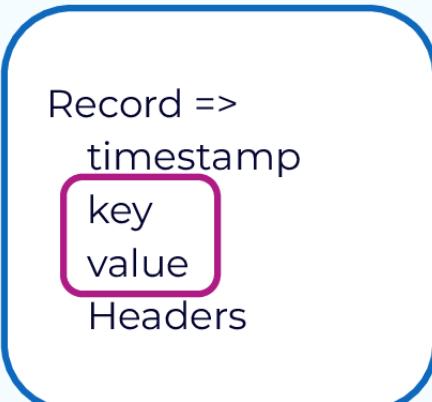
Immutable events



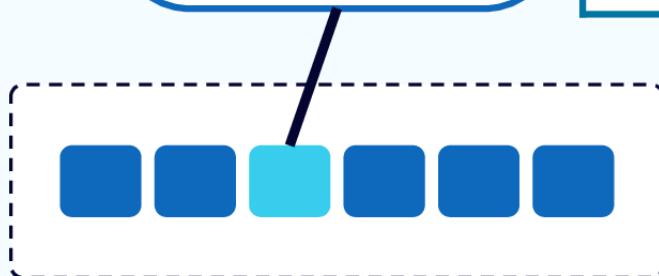
Kafka Message

- ...A message is simply an array of bytes as far as Kafka is concerned, so the data contained within it **does not have a specific format or meaning to Kafka**
- If you are approaching Kafka from a database background, you can think of this as similar to a *row* or a *record*
- A message can have an optional piece of metadata, which is referred to as a **key**. The key is also a byte array and, as with the message, has no specific meaning to Kafka
- **Keys are used when messages are to be written to partitions in a more controlled manner...**

Each Kafka message = <Key,Value>



key/ value Bytes	Area	Description
0	Magic Byte	Confluent serialization format version number; currently always 0 .
1-4	Schema ID	4-byte schema ID as returned by Schema Registry.
5...	Data	Serialized data for the specified schema format.



Event Stream

Kafka messages in batches

- For efficiency, **messages are written into Kafka in batches**.
- A **batch** is just a collection of messages, all of which are being produced to the **same topic and partition**.
- An individual round trip across the network for each message would result in excessive overhead, and collecting messages together into a batch reduces this.
- Of course, this is a trade-off between latency and throughput: **the larger the batches, the more messages that can be handled per unit of time**, but the longer it takes an individual message to propagate.
- Batches are also typically compressed, providing more efficient data transfer and storage at the cost of some processing power

Kafka Topic and partitions

- Messages in Kafka are categorized into **topics**. The closest analogies for a topic are a database *table* or a *folder* in a filesystem.
- Topics are additionally **broken down into a number of partitions**. Going back to the “*commit log*” description, a partition is a single log.
- Messages are written to it in an append-only fashion and are read in order from beginning to end.
- Note that as a topic typically has multiple partitions, there is **no guarantee of message ordering across the entire topic, just within a single partition**

Kafka Topic and partitions

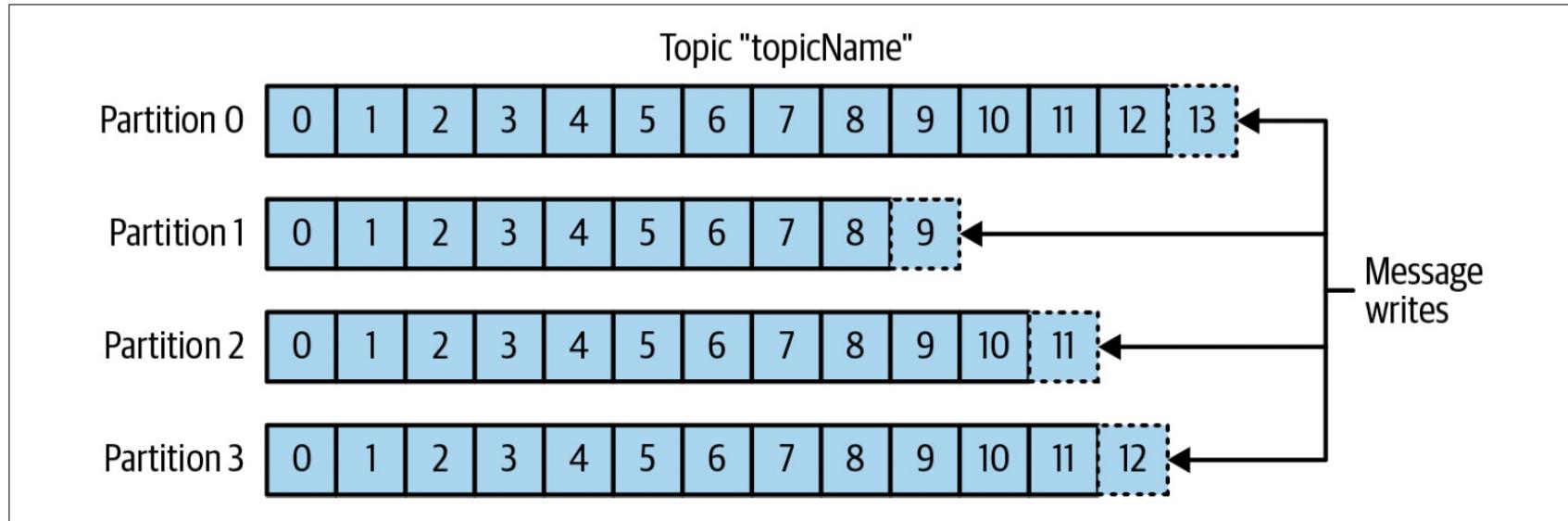


Figure 1-5. Representation of a topic with multiple partitions

Log directory

Figure 2.6. The logs directory is the base storage for messages. Each directory under /logs represents a topic partition. Filenames within the directory start with the name of the topic, followed by an underscore, which is followed by a partition number.

The logs directory is configured in the root at /logs.

/logs

 /logs/topicA_0 topicA has one partition.

 /logs/topicB_0 topicB has three partitions.

 /logs/topicB_1

 /logs/topicB_2

Listing the count of partitions for a specific topic

```
/usr/local/kafka/bin/kafka-topics.sh --describe --bootstrap-server <YourIP_or_DNS>:9092 --topic clicks
```

Topic:clicks	PartitionCount:8	ReplicationFactor:1	Configs:
Topic: clicks	Partition: 0	Leader: 0	Replicas: 0 Isr: 0
Topic: clicks	Partition: 1	Leader: 0	Replicas: 0 Isr: 0
Topic: clicks	Partition: 2	Leader: 0	Replicas: 0 Isr: 0
Topic: clicks	Partition: 3	Leader: 0	Replicas: 0 Isr: 0
Topic: clicks	Partition: 4	Leader: 0	Replicas: 0 Isr: 0
Topic: clicks	Partition: 5	Leader: 0	Replicas: 0 Isr: 0
Topic: clicks	Partition: 6	Leader: 0	Replicas: 0 Isr: 0
Topic: clicks	Partition: 7	Leader: 0	Replicas: 0 Isr: 0

Partition Number = $\text{hash(key)} \% \#Partitions(\text{topic})$

- Keys are used when messages are to be written to partitions in a more controlled manner....
- The simplest such scheme is to generate a consistent hash of the key and then select the partition number for that message by taking the result of the hash modulo the total number of partitions in the topic.
- This ensures that messages with the same key are always written to the same partition (*provided that the partition count does not change*)

Partition Number = $\text{hash(key)} \% \#Partitions(\text{topic})$

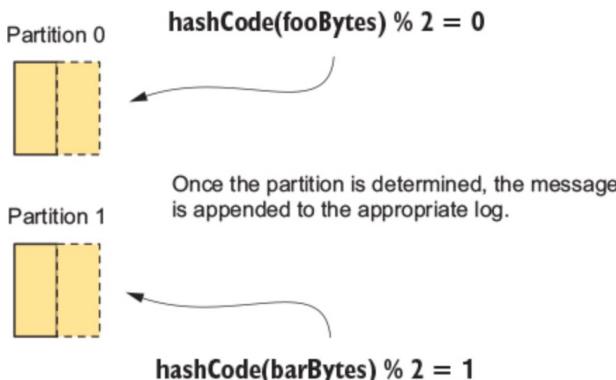
Figure 2.8. “foo” is sent to partition 0, and “bar” is sent to partition 1. You obtain the partition by hashing the bytes of the key, modulus the number of partitions.

Incoming messages:

```
{foo, message data}  
{bar, message data}
```

Message keys are used to determine which partition the message should go to. These keys are not null.

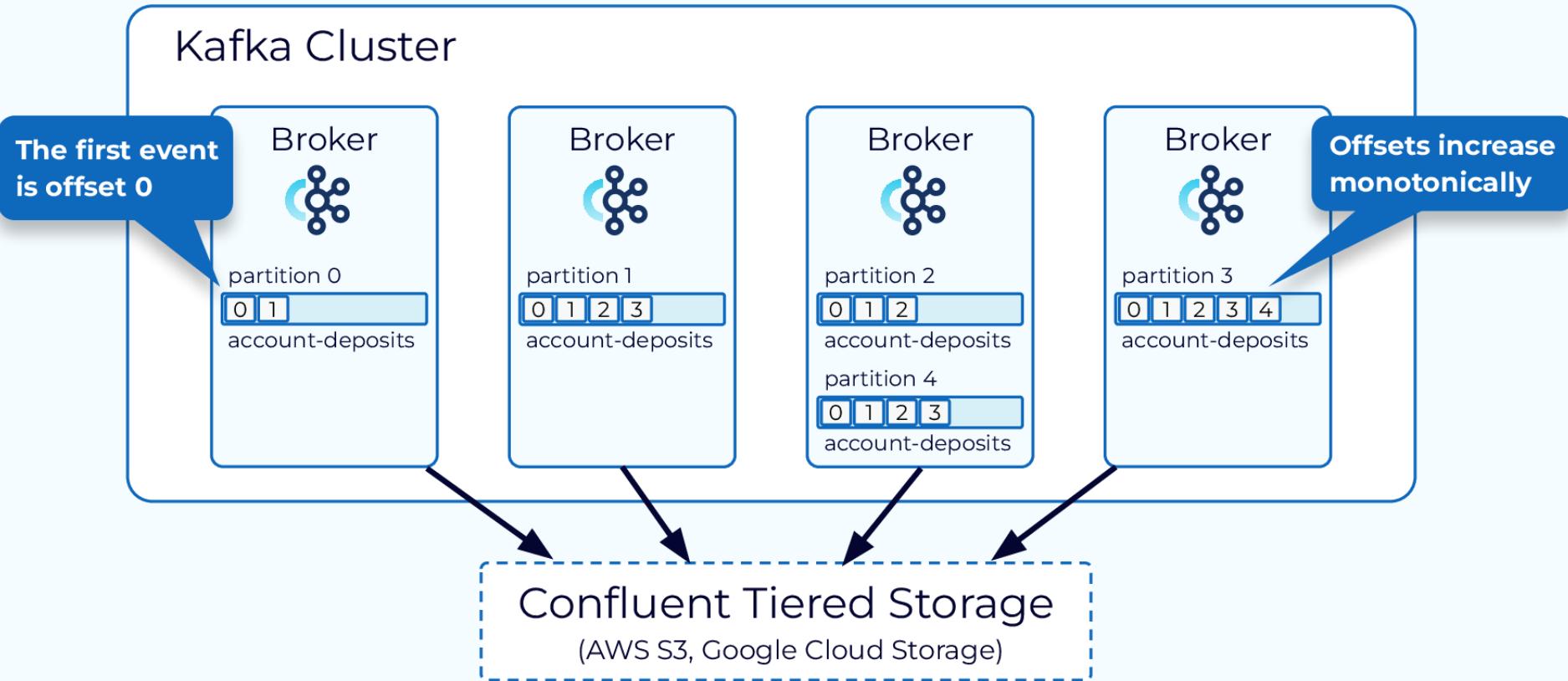
The bytes of the key are used to calculate the hash.



Message Offset

- The offset—an integer value that continually increases—is another piece of metadata that Kafka adds to each message as it is produced
- **Each message in a given partition has a unique offset**, and the following message has a greater offset (though not necessarily monotonically greater)

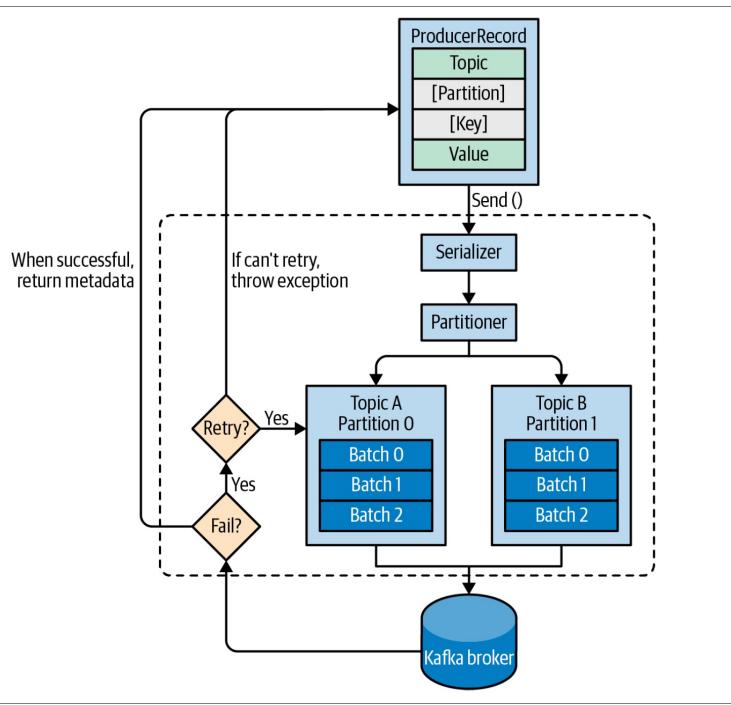
Message offset increasing monotonically



Kafka Producer

- Producers create **new messages**
- **A message will be produced to a specific topic.** By default, the producer will **balance** messages over all partitions of **a topic evenly**.
In some cases, the producer will direct messages to specific partitions. This is typically done using the message key and a partitioner that will generate a hash of the key and map it to a specific partition
- This ensures that all messages produced with a given key will get written to the same partition. *The producer could also use a custom partitioner that follows other business rules for mapping messages to partitions*

Producer sending message to the broker



Sending policies

- **Fire-and-forget**

Sending a message to the server and do not really care if it arrives successfully or not. Most of the time, it will arrive successfully, since Kafka is highly available, and the producer will retry automatically. However, some messages will get lost using this method.

- **Synchronous send**

Sending a message and testing the acknowledgement: the `send()` method returns a future object, and the sender can call `get()` to wait on the future and see if send was successful.

- **Asynchronous send**

Calling the `send()` method with a callback function, which gets triggered when it receives a response from the Kafka broker. **Whenever the synchronous responses take too long, or no error processing need to be done.**

Figure 3-1. High-level overview of Kafka producer components

JAVA Source code for fire & forget sending policy

```
1 import java.util.Properties;
2 import org.apache.kafka.clients.producer.Callback;
3 import org.apache.kafka.clients.producer.KafkaProducer;
4 import org.apache.kafka.clients.producer.Producer;
5 import org.apache.kafka.clients.producer.ProducerConfig;
6 import org.apache.kafka.clients.producer.ProducerRecord;
7 import org.apache.kafka.clients.producer.RecordMetadata;
8 import org.apache.kafka.common.serialization.LongSerializer;
9 import org.apache.kafka.common.serialization.StringSerializer;
10 public class ClientProducerKafka {
11     Run | Debug
12     public static void main(String[] args) {
13
14         Properties kafkaProps = new Properties();
15         kafkaProps.put(key: "bootstrap.servers", value: "YOURAWSIP:9092");
16         kafkaProps.put(key: "key.serializer", value: "org.apache.kafka.common.serialization.StringSerializer");
17         kafkaProps.put(key: "value.serializer", value: "org.apache.kafka.common.serialization.StringSerializer");
18         KafkaProducer<String, String> producer = new KafkaProducer<String, String>(kafkaProps);
19
20         //Fire-and-forget
21         System.out.println(x: "Fire-and-forget starting...");
22         ProducerRecord<String, String> record =
23             new ProducerRecord<>("NAMEOFTOPIC", "KEY", "MESSAGE");
24         try { producer.send(record); }
25         catch (Exception e) { e.printStackTrace(); }
26         System.out.println(x: "Fire-and-forget stopped.");
27     }
28 }
```

JAVA Source code for synchronous sending policy

```
1 import java.util.Properties;
2 import org.apache.kafka.clients.producer.Callback;
3 import org.apache.kafka.clients.producer.KafkaProducer;
4 import org.apache.kafka.clients.producer.Producer;
5 import org.apache.kafka.clients.producer.ProducerConfig;
6 import org.apache.kafka.clients.producer.ProducerRecord;
7 import org.apache.kafka.clients.producer.RecordMetadata;
8 import org.apache.kafka.common.serialization.LongSerializer;
9 import org.apache.kafka.common.serialization.StringSerializer;
10 public class ClientProducerKafka {
11     Run | Debug
12     public static void main(String[] args) {
13         Properties kafkaProps = new Properties();
14         kafkaProps.put(key: "bootstrap.servers", value: "YOURAWSIP:9092");
15         kafkaProps.put(key: "key.serializer", value: "org.apache.kafka.common.serialization.StringSerializer");
16         kafkaProps.put(key: "value.serializer", value: "org.apache.kafka.common.serialization.StringSerializer");
17         KafkaProducer<String, String> producer = new KafkaProducer<String, String>(kafkaProps);
18
19         //Synchronous send
20         System.out.println(x: "Synchronous send starting...");
21         ProducerRecord<String, String> record2 =
22             new ProducerRecord<>("NAMEOFTOPIC", "KEY", "MESSAGE2");
23         try { System.out.println("Result get =" + producer.send(record2).get()); }
24         catch (Exception e) {e.printStackTrace(); }
25         System.out.println(x: "Synchronous stopped.");
26     }
27 }
28 }
```

```
private class DemoProducerCallback implements Callback { ①
    @Override
    public void onCompletion(RecordMetadata recordMetadata, Exception e) {
        if (e != null) {
            e.printStackTrace(); ②
        }
    }
}

ProducerRecord<String, String> record =
    new ProducerRecord<>("CustomerCountry", "Biomedical Materials", "USA"); ③
producer.send(record, new DemoProducerCallback()); ④
```

- ➊ To use callbacks, you need a class that implements the `org.apache.kafka.clients.producer.Callback` interface, which has a single function—`onCompletion()`.
- ➋ If Kafka returned an error, `onCompletion()` will have a nonnull exception. Here we “handle” it by printing, but production code will probably have more robust error handling functions.
- ➌ The records are the same as before.
- ➍ And we pass a `Callback` object along when sending the record.

JAVA Source code for asynchronous sending policy

Kafka Consumer

- Consumers **read messages**
- The consumer **subscribes to one or more topics** and reads the messages in the **order** in which they were produced **to each partition**.
- The consumer keeps track of which messages it has already consumed by **keeping track of the offset of messages**
- By storing the next possible offset for each partition a consumer can stop and restart without losing its place.

Kafka Consumer Group

- Consumers work as **part of a consumer group**, which is one or more consumers that work together to consume a topic. The group ensures that **each partition is only consumed by one member**.
- In this way, consumers can **horizontally scale** to consume topics with a large number of messages. Additionally, **if a single consumer fails**, the remaining members of the group will **reassign the partitions being consumed** to take over for the missing member: **Rebalance**

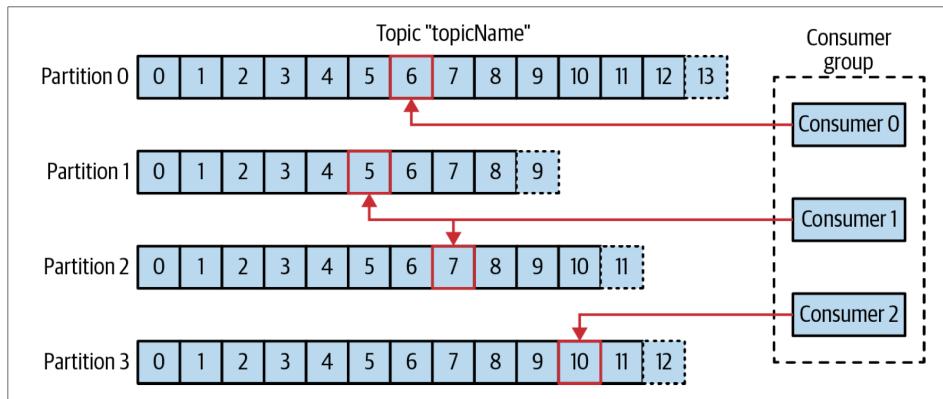


Figure 1-6. A consumer group reading from a topic

Here are three consumers in a single group consuming a topic. Two of the consumers are working from one partition each, while the third consumer is working from two partitions.

The mapping of a consumer to a partition is often called *ownership* of the partition by the consumer.

On Kafka Consumer Group rebalance

- On rebalancing, Kafka needs to know the offset where the actual consumers were reading
- How to communicate the offset to Kafka?
 - Automatic commit:

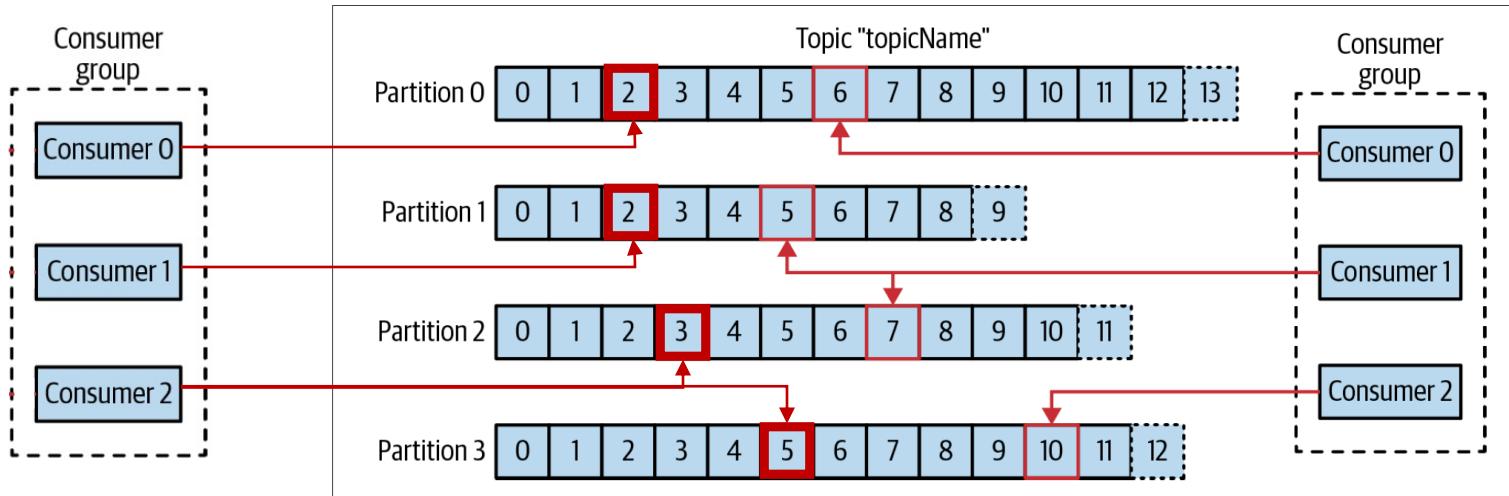
```
enable.auto.commit=true;
```

then every **five seconds** the consumer will commit the largest offset your client receive from `poll()`.
 - Or, explicitly Commit current Offset

```
try
{
    consumer.commitSync();
}
catch (CommitFailedException e){log.error("commit failed",e)}
```

Kafka Consumer Group

- Each consumer group have its **own offset**
- Consumer groups remember the offsets where they left off
- *Example, 2 consumer groups for the same topic with different offsets. Both consumer groups are consuming the same messages:*



Listing the consumer-ids from a specific consumer group

```
sudo /usr/local/kafka/bin/kafka-consumer-groups.sh --bootstrap-server <YourIP_or_DNS>:9092 -describe -group
console-consumer-28356
```

TOPIC	PARTITION	CURRENT-OFFSET	LOG-END-OFFSET	LAG	CONSUMER-ID	HOST	CLIENT-ID
clicks	1	-	0	-	consumer-1-4924ed20-7fc8-46e4-909a-35c71d5a79fc	/194.210.61.144	consumer-1
clicks	4	-	0	-	consumer-1-4924ed20-7fc8-46e4-909a-35c71d5a79fc	/194.210.61.144	consumer-1
clicks	7	-	0	-	consumer-1-4924ed20-7fc8-46e4-909a-35c71d5a79fc	/194.210.61.144	consumer-1
clicks	0	-	1	-	consumer-1-4924ed20-7fc8-46e4-909a-35c71d5a79fc	/194.210.61.144	consumer-1
clicks	5	-	0	-	consumer-1-4924ed20-7fc8-46e4-909a-35c71d5a79fc	/194.210.61.144	consumer-1
clicks	2	-	1	-	consumer-1-4924ed20-7fc8-46e4-909a-35c71d5a79fc	/194.210.61.144	consumer-1
clicks	3	-	1	-	consumer-1-4924ed20-7fc8-46e4-909a-35c71d5a79fc	/194.210.61.144	consumer-1
clicks	6	-	0	-	consumer-1-4924ed20-7fc8-46e4-909a-35c71d5a79fc	/194.210.61.144	consumer-1

JAVA Source code for consuming from a topic

Polling mechanism

```
1 import java.util.Collections;
2 import java.util.Properties;
3 import org.apache.kafka.clients.consumer.ConsumerRecord;
4 import org.apache.kafka.clients.consumer.ConsumerRecords;
5 import org.apache.kafka.clients.consumer.KafkaConsumer;
6 public class ClientConsumerKafka {
7     Run | Debug
8     public static void main(String[] args) {
9         Properties props = new Properties();
10        props.put(key: "bootstrap.servers", value: "yourAWSIP:9092");
11        props.put(key: "group.id", value: "CountryCounter");
12        props.put(key: "key.deserializer", value: "org.apache.kafka.common.serialization.StringDeserializer");
13        props.put(key: "value.deserializer", value: "org.apache.kafka.common.serialization.StringDeserializer");
14        KafkaConsumer < String, String > consumer = new KafkaConsumer < String, String > (props);
15        consumer.subscribe(Collections.singletonList(o: "NAMEOFTOPIC"));
16        try {
17            while (true) {
18                /* consumers must keep polling Kafka or they will be considered
19                   dead and the partitions they are consuming will be handed to another
20                   consumer in the group to continue consuming. The parameter we pass, poll(),
21                   is a timeout interval and controls how long poll() will block if data is not available
22                   in the consumer buffer. If this is set to 0, poll() will return immediately;
23                   otherwise, it will wait for the specified number of milliseconds for data to arrive
24                   from the broker. */
25                ConsumerRecords < String, String > records = consumer.poll(100);
26                for (ConsumerRecord < String, String > record: records) {
27                    System.out.printf(format: "topic = %s, partition = %s, offset = %d, customer = %s, country = %s\n",
28                                     record.topic(), record.partition(), record.offset(), record.key(), record.value());
29                }
30            } finally {
31                consumer.close();
32            }
33        }
34    }
```

On Kafka Partitions and Consumer Groups

- From a performance point of view, it is the number of **partitions** that matters.
- But since each topic in Kafka has at least one partition, if you have N topics, you inevitably have at least N partitions
- However, keep in mind that, partitions have costs (latency, more memory, more file descriptors...)
- Create a **new consumer group** for each application that needs all the messages from **one or more topics**
- Add consumers to an existing consumer group to scale reading and processing messages from topics. Obviously, it is **bounded by the number of partitions in the topic**
- The reads from a single partition maintain the order in which the messages were produced but within a **multiple partition Topic global order is not guaranteed**

Kafka Broker

- A single Kafka server is called a **broker**.
- The broker receives messages from producers, assigns offsets to them, and writes the messages to storage on disk.
- It also services consumers, responding to fetch requests for partitions and responding with the messages that have been published.
- Depending on the specific hardware and its performance characteristics, a single broker can easily **handle thousands of partitions and millions of messages per second**.
- A broker can define a **retention period** for data which can be configured by each Topic: a *given period* and a *given size*

A simple Kafka Cluster

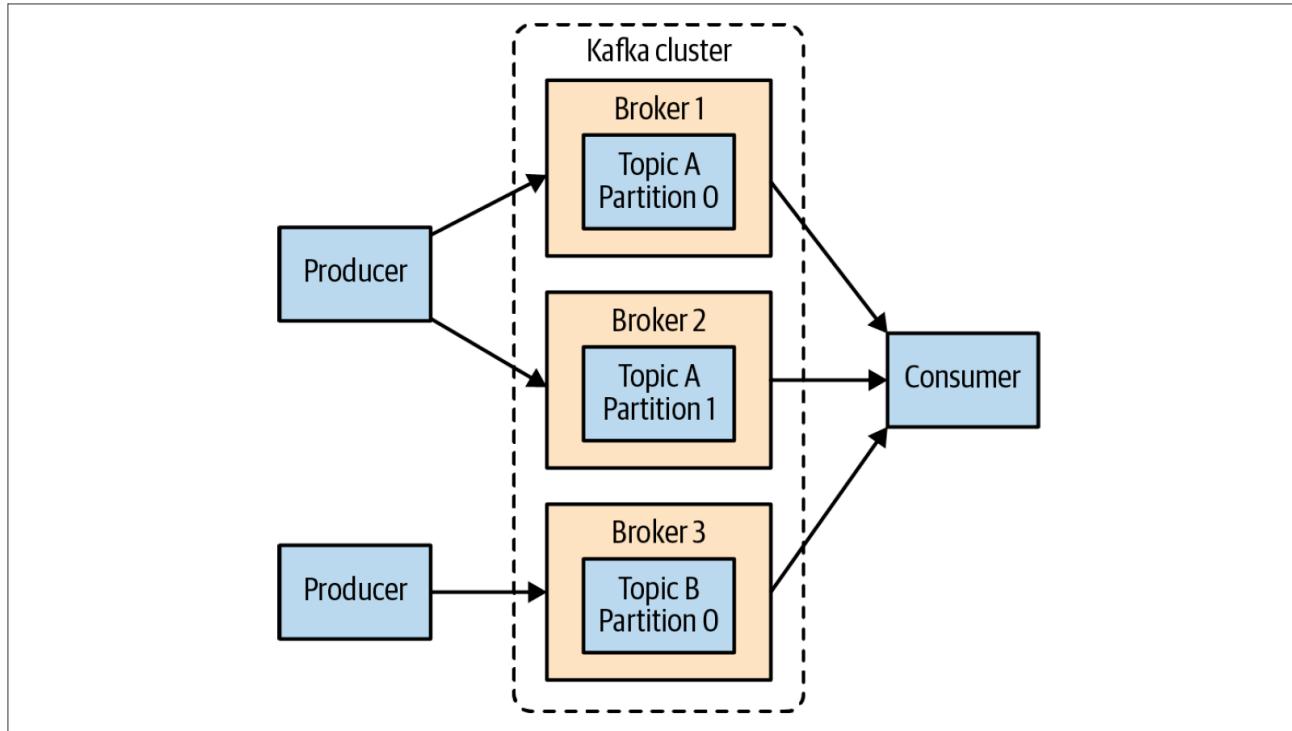


Figure 2-2. A simple Kafka cluster

Kafka Cluster

- Kafka brokers are designed to operate as part of a cluster.
- Within a cluster of brokers, one broker will also function as the **cluster controller** (**elected** automatically from the live members of the cluster).
- The controller is responsible for administrative operations, including assigning partitions to brokers and monitoring for broker failures.
- A partition is owned by a single broker in the cluster, and that broker is called the **leader of the partition**. A replicated partition is assigned to additional brokers, called **followers** of the partition (*next figure*).
- Replication provides **redundancy** of messages in the partition, such that one of the followers can take over leadership if there is a broker failure. All producers must connect to the leader in order to publish messages, but consumers may fetch from either the leader or one of the followers.

Replication of partitions in a cluster

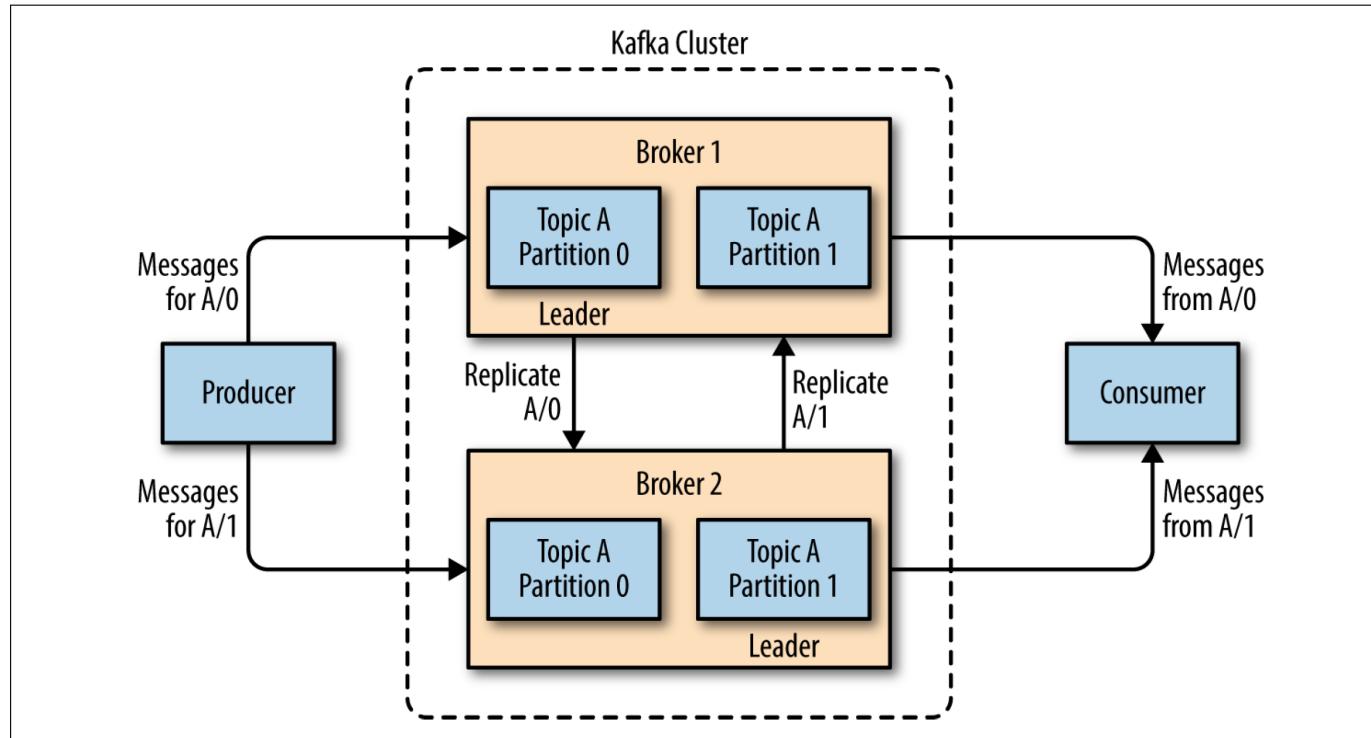
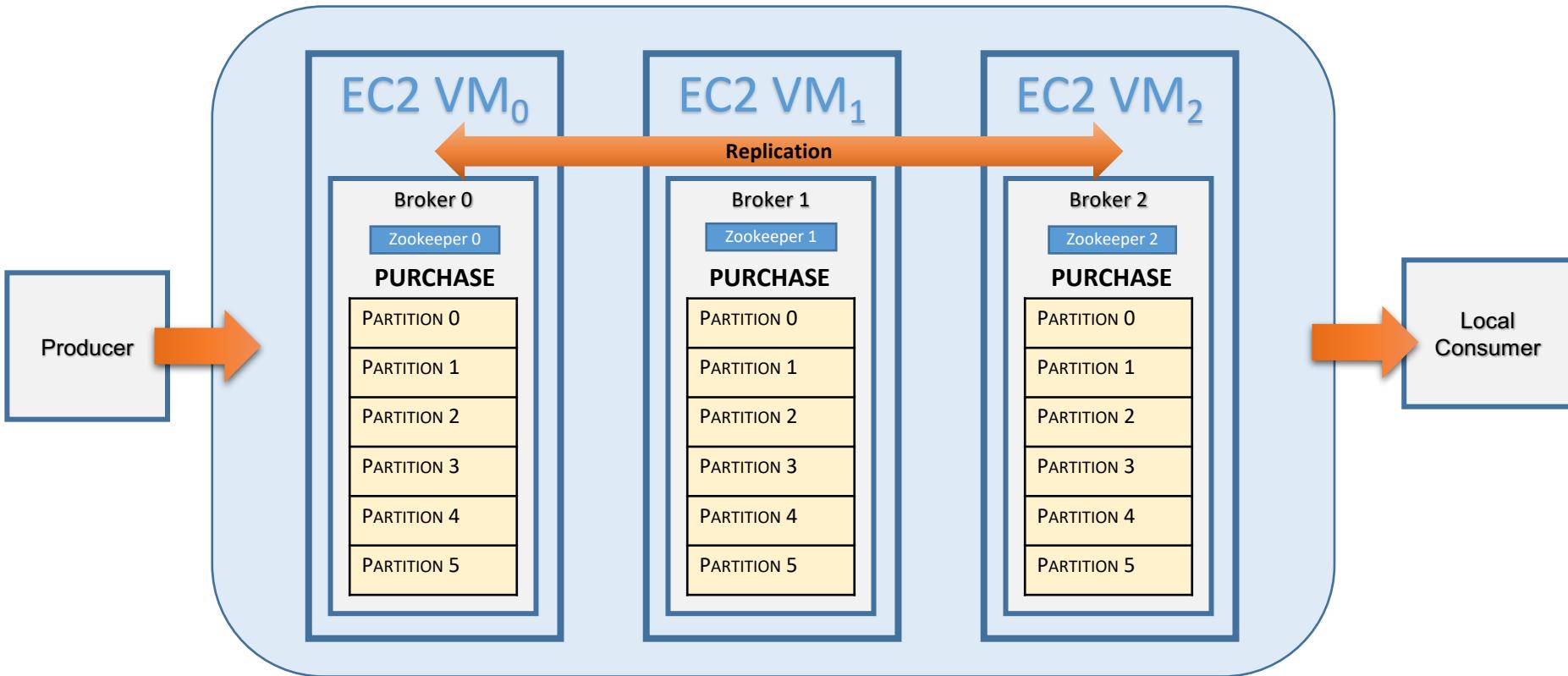
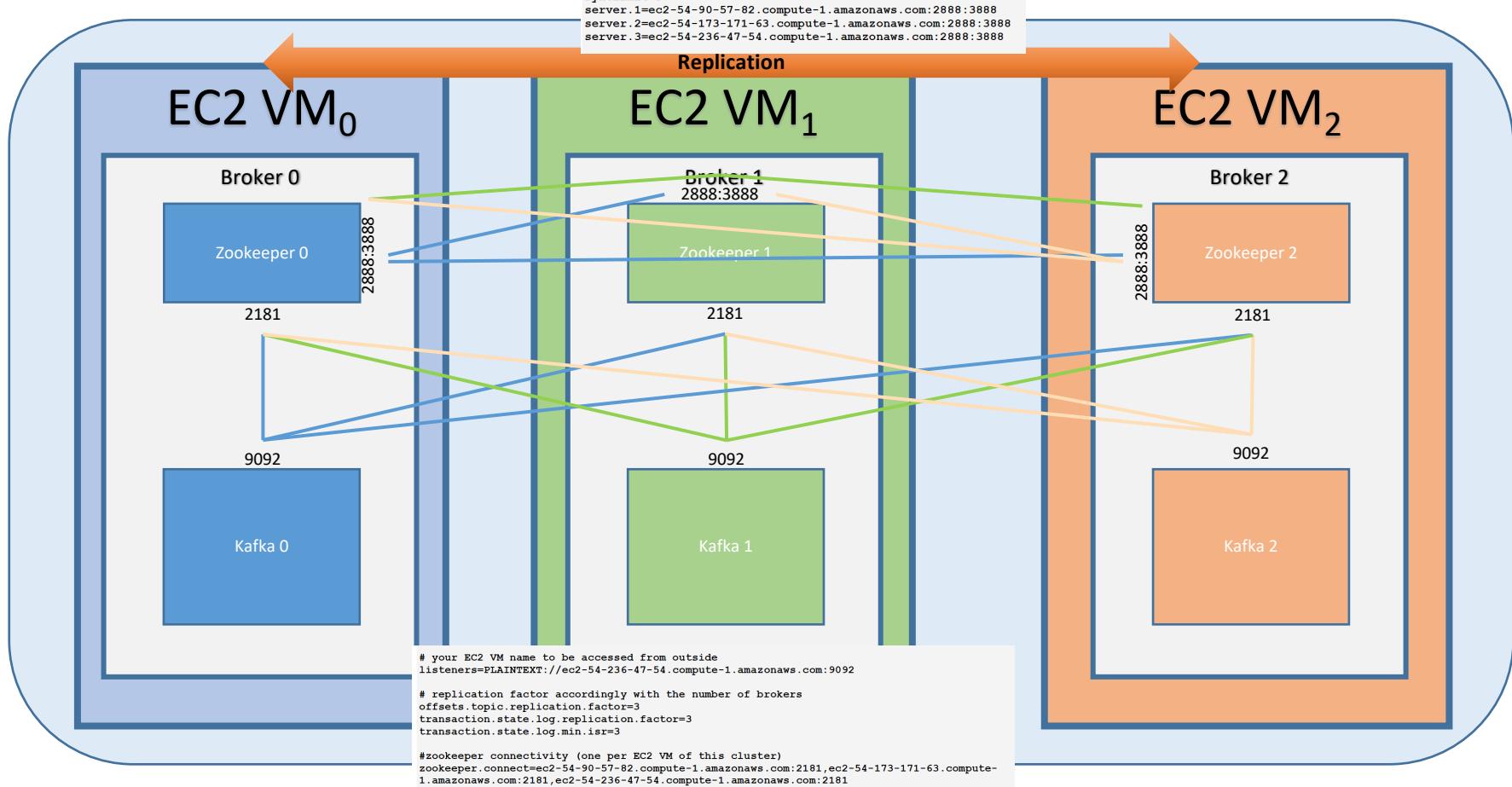


Figure 1-7. Replication of partitions in a cluster

Kafka Cluster deployment with 3 brokers



Implementation of a Kafka Cluster



in kafka is it possible to consume a message that is not propagated to all brokers?



In Apache Kafka, a message is typically written to a topic and then replicated across multiple brokers for fault tolerance and high availability. By default, a message needs to be written to all in-sync replicas (ISRs) of a partition before it can be considered committed and available for consumption.



However, it is possible to configure Kafka to allow consumers to read messages that have not been fully replicated to all brokers. This is called the "unclean leader election" mode, which is disabled by default for data consistency reasons.

When unclean leader election is enabled, Kafka may allow a replica that is not fully up-to-date to become the leader of a partition. In this case, the leader may serve messages that have not yet been replicated to all brokers, and consumers may be able to consume these messages from the leader before they are propagated to all replicas.

It's worth noting that enabling unclean leader election can compromise data consistency and increase the risk of data loss, so it should be used with caution and only in certain scenarios where data consistency can be relaxed.

Broker failure test

Starting a cluster with 2 brokers

```
/usr/local/kafka/bin/kafka-topics.sh --describe --bootstrap-server <YourIP_or_DNS>:9092 --topic events
```

Topic:events PartitionCount:4	ReplicationFactor:2 Configs:
Topic: events Partition: 0	Leader: 0 Replicas: 0,1 Isr: 0,1
Topic: events Partition: 1	Leader: 1 Replicas: 1,0 Isr: 1,0
Topic: events Partition: 2	Leader: 0 Replicas: 0,1 Isr: 0,1
Topic: events Partition: 3	Leader: 1 Replicas: 1,0 Isr: 1,0

in-sync replicas (ISRs)

Stopping broker 0

```
/usr/local/kafka/bin/kafka-topics.sh --describe --bootstrap-server <YourIP_or_DNS>:9092 --topic events
```

Topic:events PartitionCount:4	ReplicationFactor:2 Configs:
Topic: events Partition: 0	Leader: 1 Replicas: 0,1 Isr: 1
Topic: events Partition: 1	Leader: 1 Replicas: 1,0 Isr: 1
Topic: events Partition: 2	Leader: 1 Replicas: 0,1 Isr: 1
Topic: events Partition: 3	Leader: 1 Replicas: 1,0 Isr: 1

Best size for Kafka Cluster?

- **Criteria 1:** Data storage available on each kafka broker versus overall storage need?
 - E.g.: retain 10TB and each broker store 2TB = cluster with 5 brokers
- **Criteria 2:** Handling requests capacity?
 - E.g.: what is the capacity of the network interfaces, and can they handle the client traffic if there are multiple consumers of the data or if the traffic is not consistent over the retention period of the data. Consider the bursts of traffic during peak times

How many Kafka partitions?

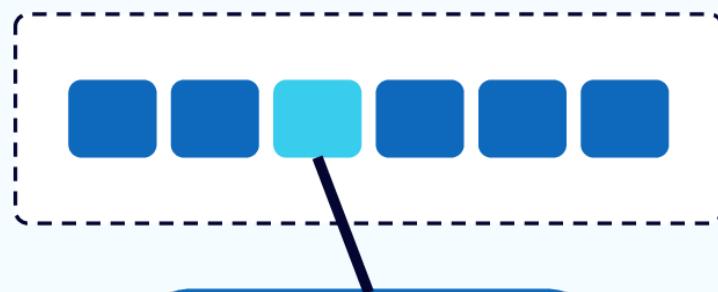
- Producers are typically much faster than consumers, then always calculate for consumers' expectations
- What is the maximum throughput expected to achieve when consuming from a single partition?
 - Considering, always one consumer reading from a partition, so if you know that your slower consumer writes the data to a database that never handles more than 50 MB per second from each thread writing to it, then you know you are **limited to 50MB throughput** when consuming from a partition
- If you are sending messages to partitions based on keys, adding partitions later **can be very challenging**, so calculate throughput based on your expected future usage, not the current usage
- Avoid overestimating, as each partition uses memory and other resources on the broker and will increase the time for leader elections

Event Stream Processing

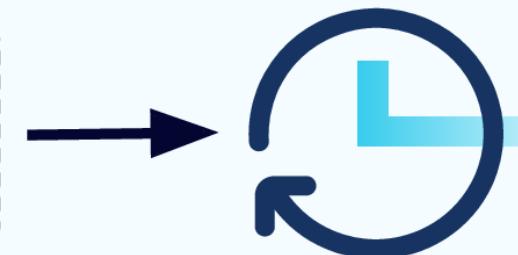
Event Source



Event Stream



Event Processing
Application



Record =>
timestamp
key
value
Headers

Kafka Streams

- What is a data stream (also called an event stream or streaming data)?
- First and foremost, a data stream is an abstraction representing an **unbounded dataset**.
- Unbounded means infinite and ever growing. The dataset is unbounded because over time, **new records keep arriving**
- Note that this simple model (a stream of events) can be used to represent just about every business activity we care to analyze. We can look at a stream of credit card transactions, stock trades, package deliveries, network events going through a switch, events reported by sensors in manufacturing equipment, emails sent, moves in a game, etc. The list of examples is endless because pretty much everything can be seen as a sequence of events.

Kafka Streams properties

- **Unbounded**
- Event streams are **ordered**
- **Immutable** data records
- Event streams are **replayable** - to correct errors, try new methods of analysis, or perform audits

Kafka Streams processing

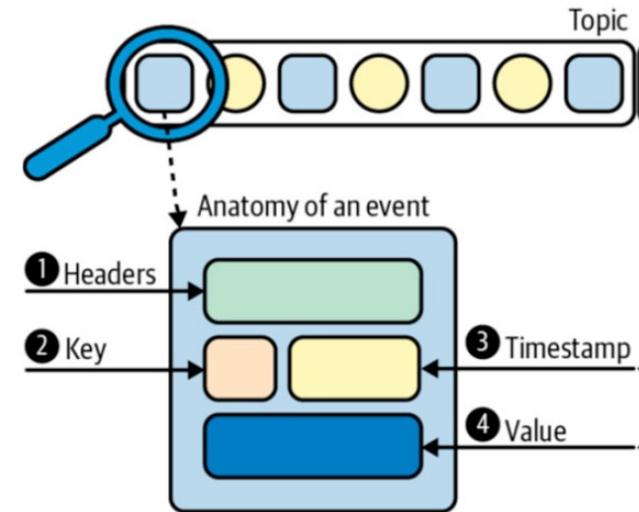
- Continuous and nonblocking: Stream processing fills the gap between the request-response world, where we wait for events that take two milliseconds to process, and the batch processing world, where data is processed once a day and takes eight hours to complete.
- Most business processes don't require an immediate response within milliseconds but can't wait for the next day either. Most business processes happen continuously, and as long as the business reports are updated continuously and the line of business apps can continuously respond, the processing can proceed without anyone waiting for a specific response within milliseconds.
- Business processes such as alerting on suspicious credit transactions or network activity, adjusting prices in real-time based on supply and demand, or tracking deliveries of packages are all a natural fit for continuous but nonblocking processing.

Topology

- A stream processing application includes **one or more processing topologies**.
- A processing topology starts with **one or more source streams** that are passed through a graph of stream processors connected through event streams, until results are written to one or more sink streams.
- Each stream processor is a computational step applied to the stream of events in order to transform the events.

Time

- **Event time** - This is the time the events we are tracking occurred and the record was created
- **Log append time** - This is the time the event arrived at the Kafka broker and was stored there, also called ingestion time
- **Processing time** - This is the time at which a stream processing application received the event in order to perform some calculation



State

- **To keep track of more information**— how many events of each type did we see this hour, all events that require joining, sums, averages, etc.
- **Local or internal state** - State that is accessible only by a specific instance of the stream processing application. This state is usually maintained and managed with an embedded, in-memory database running within the application.
 - The advantage of local state is that it is extremely fast.
 - The disadvantage is that we are limited to the amount of memory available
- **External state** - State that is maintained in an external data store, often a NoSQL system like Cassandra.
 - The advantages of an external state are its virtually unlimited size and the fact that it can be accessed from multiple instances of the application or even from different applications.
 - The downside is the extra latency and complexity introduced with an additional system, as well as availability—the application needs to handle the possibility that the external system is not available

Table -> Stream

- To convert a table to a stream, we need to capture the changes that modify the table.
- Take all those insert, update, and delete events and store them in a stream. Most databases offer change data capture (CDC) solutions for capturing these changes, and there are many Kafka connectors that can pipe those changes into Kafka where they will be available for stream processing.

Stream -> Table

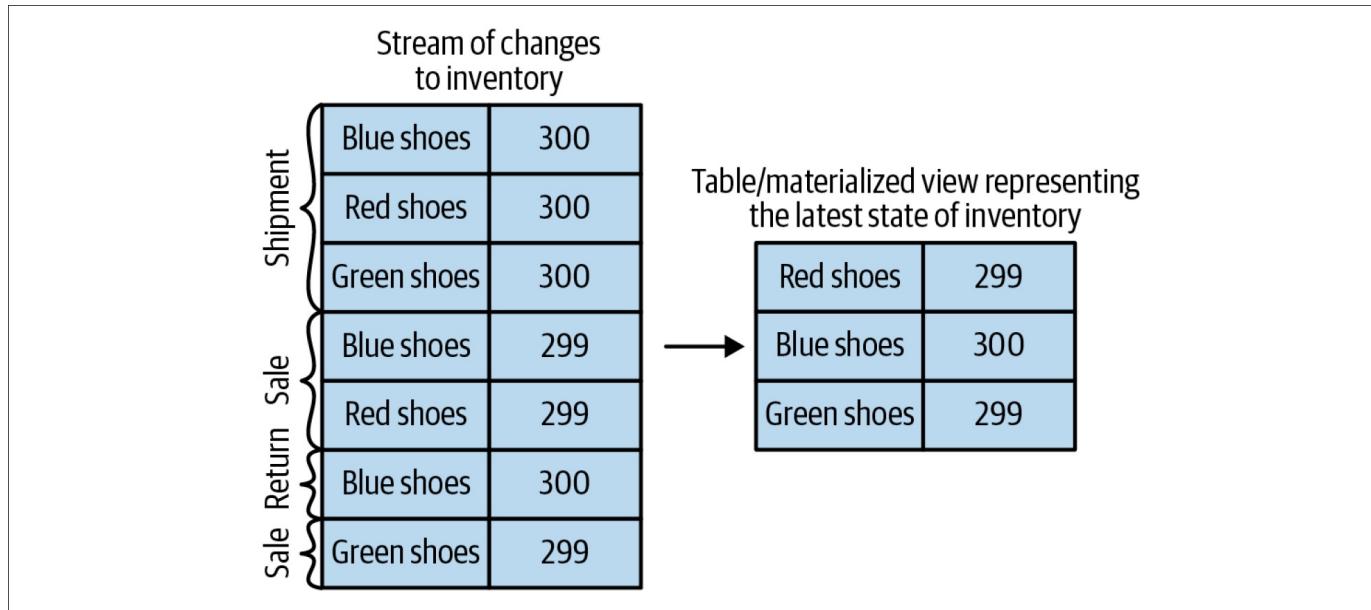
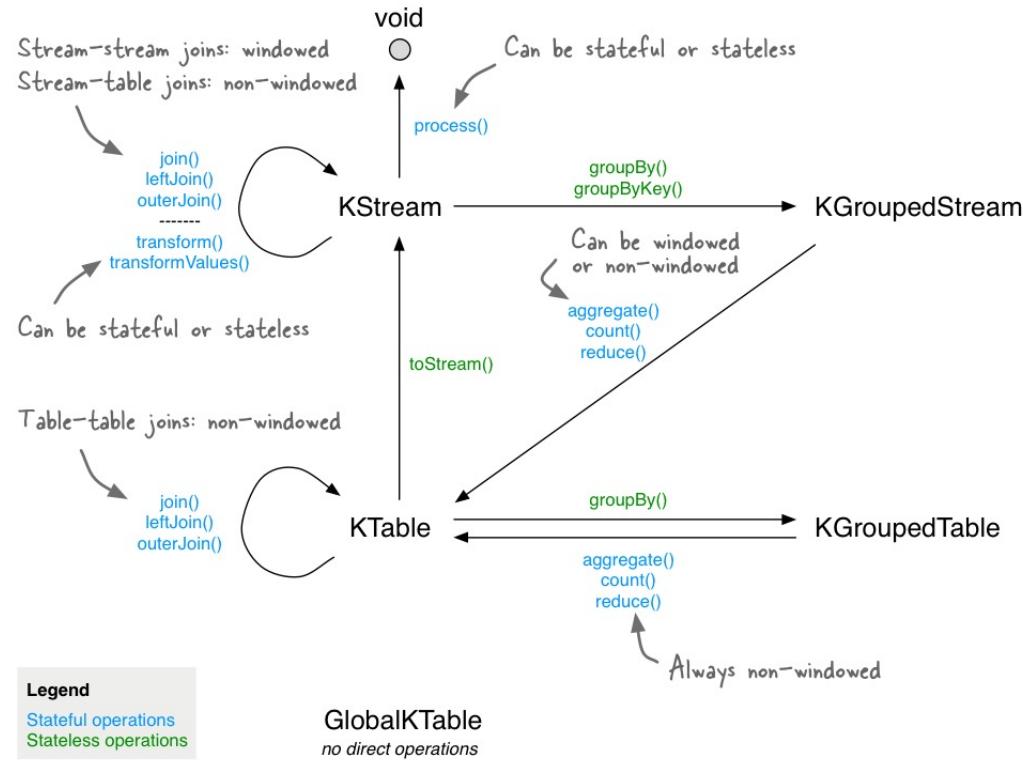


Figure 14-1. Materializing inventory changes

KStreams-KTable duality



Time Windows

Most operations on streams are windowed operations, operating on slices of time: moving averages, top products sold this week, 99th percentile load on the system, etc.

When calculating moving averages, we want to know:

- Size of the window
- How often the window moves (advance interval)
- How long the window remains updatable (grace period)

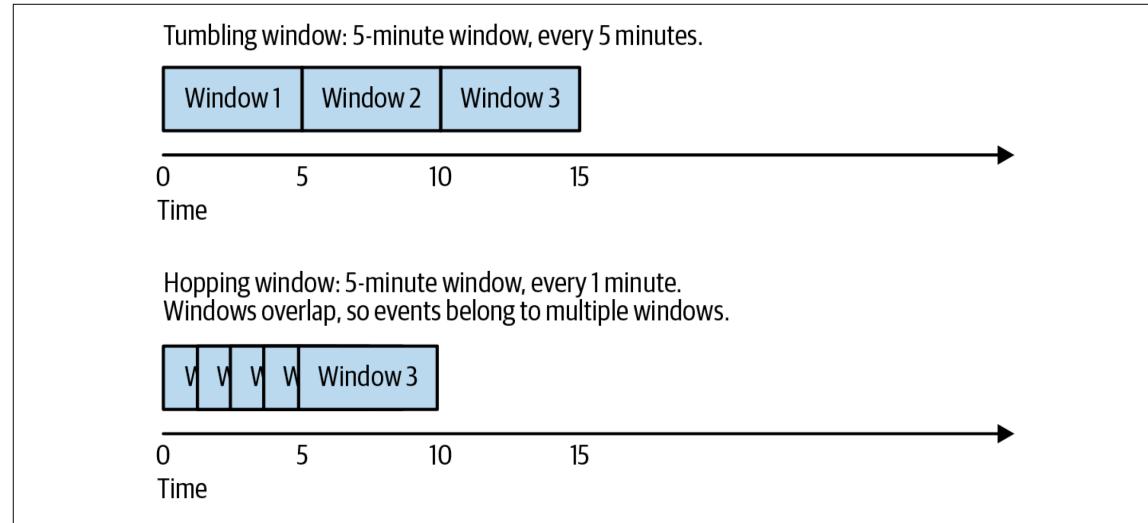


Figure 14-2. Tumbling window versus hopping window

Stream Processing Design Patterns

Single-Event Processing

handled with a simple producer and consumer

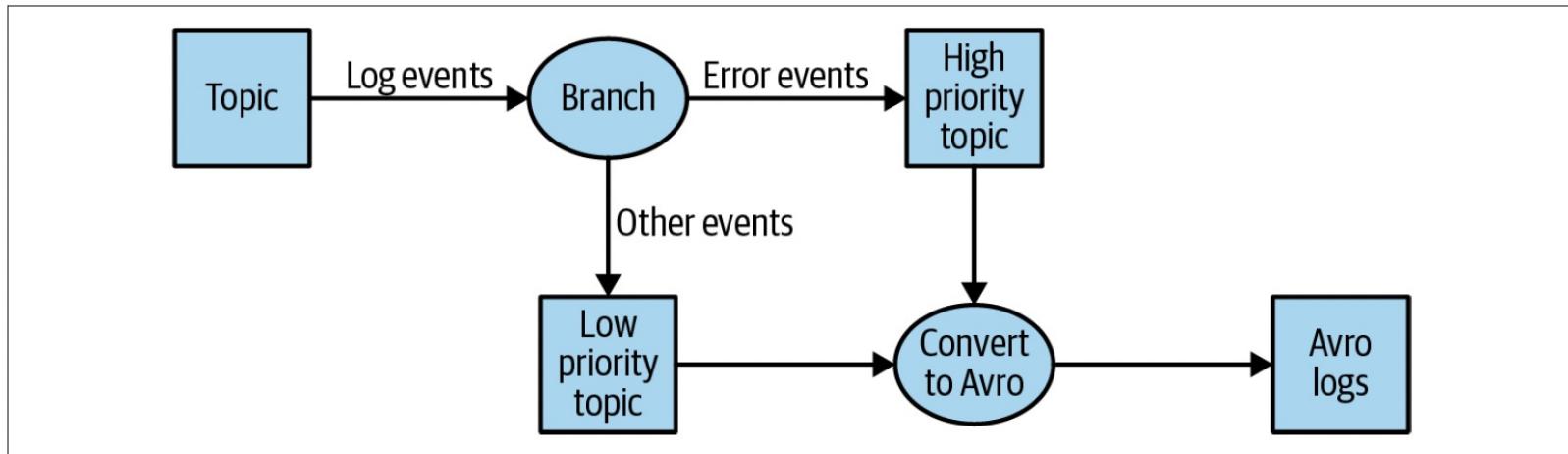
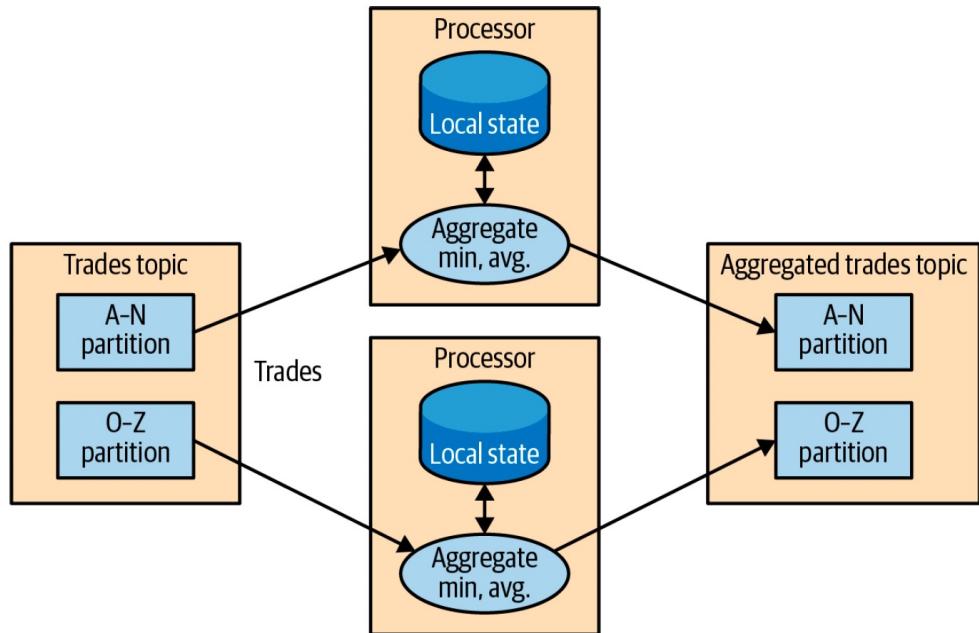


Figure 14-3. Single-event processing topology

Stream Processing Design Patterns

Processing with Local State



done using local state (rather than a shared state)

...local state is stored in-memory using embedded RocksDB, which also persists the data to disk for quick recovery after restarts

Figure 14-4. Topology for event processing with local state

Stream Processing Design Patterns

Multiphase Processing/Repartitioning

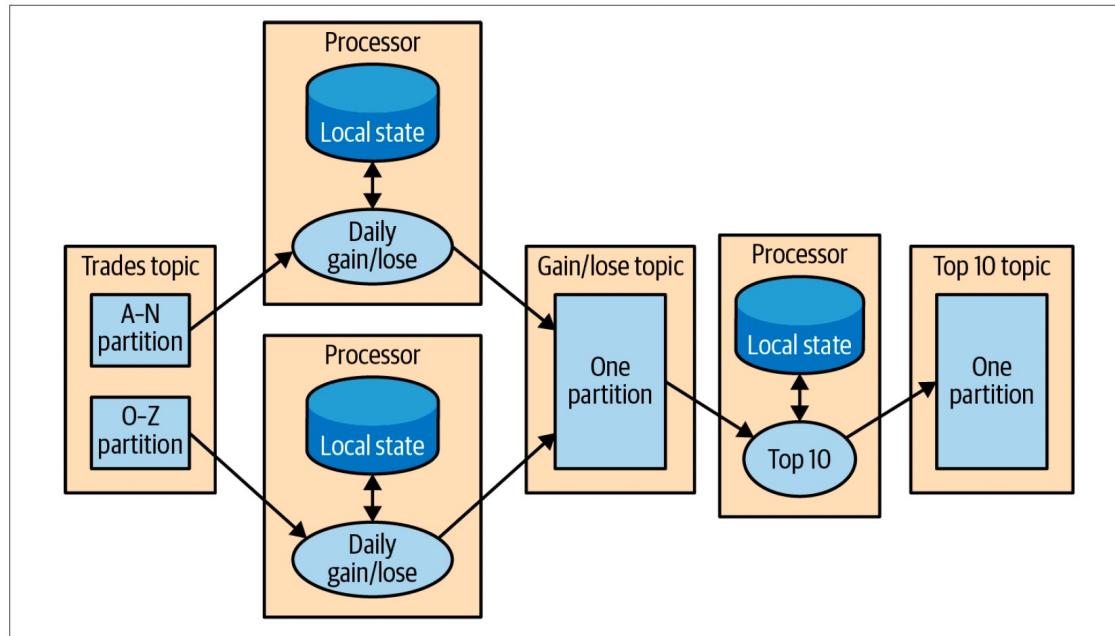


Figure 14-5. Topology that includes both local state and repartitioning steps

First, we calculate the daily gain/loss for each stock symbol. We can do this on each instance with a local state.

Then we write the results to a new topic with a single partition. This partition will be read by a single application instance that can then find the top 10 stocks for the day.

The second topic, which contains just the daily summary for each stock symbol, is obviously much smaller with significantly less traffic than the topics that contain the trades themselves, and therefore it can be processed by a single instance of the application

WordCount Example

using JAVA lambda

```

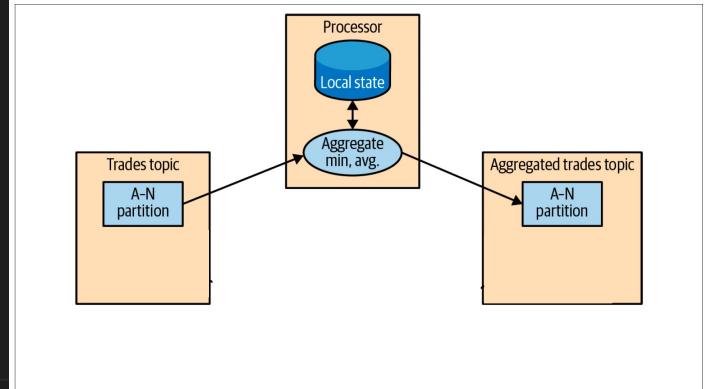
1 package com.shapira.examples.streams.wordcount;
2 import org.apache.kafka.clients.CommonClientConfigs;
3 import org.apache.kafka.clients.consumer.ConsumerConfig;
4 import org.apache.kafka.common.serialization.Serdes;
5 import org.apache.kafka.streams.KafkaStreams;
6 import org.apache.kafka.streams.KeyValue;
7 import org.apache.kafka.streams.StreamsConfig;
8 import org.apache.kafka.streams.KStream;
9 import org.apache.kafka.streams.StreamsBuilder;
10 import java.util.Arrays;
11 import java.util.Properties;
12 import java.util.regex.Pattern;
13
14 public class WordCountExample {
    Run | Debug
15     public static void main(String[] args) throws Exception{
        Properties props = new Properties();
        props.put(StreamsConfig.APPLICATION_ID_CONFIG, value: "wordcount");
        props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, value: "ec2-52-207-190-101.compute-1.amazonaws.com:9092");
        props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName());
        props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName());
        props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, value: "earliest");
        StreamsBuilder builder = new StreamsBuilder();
        KStream<String, String> source = builder.stream("wordcount-input");

        final Pattern pattern = Pattern.compile(regex: "\\W+");
        KStream<String, String> counts = source.flatMapValues(value-> Arrays.asList(pattern.split(value.toLowerCase())))
            .map((key, value) -> new KeyValue<Object, Object>(value, value))
            .filter((key, value) -> (!value.equals(obj: "the")))
            .groupByKey()
            .count().mapValues(value->Long.toString(value)).toStream();
        counts.to("wordcount-output");
    }

    KafkaStreams streams = new KafkaStreams(builder.build(), props);

    streams.cleanUp();
    streams.start();
    Thread.sleep(milliseconds: 5000L);
    streams.close();
}

```



Q&A





TÉCNICO LISBOA

Enterprise Integration

(Micro)Services

Prof. Sérgio Guerreiro

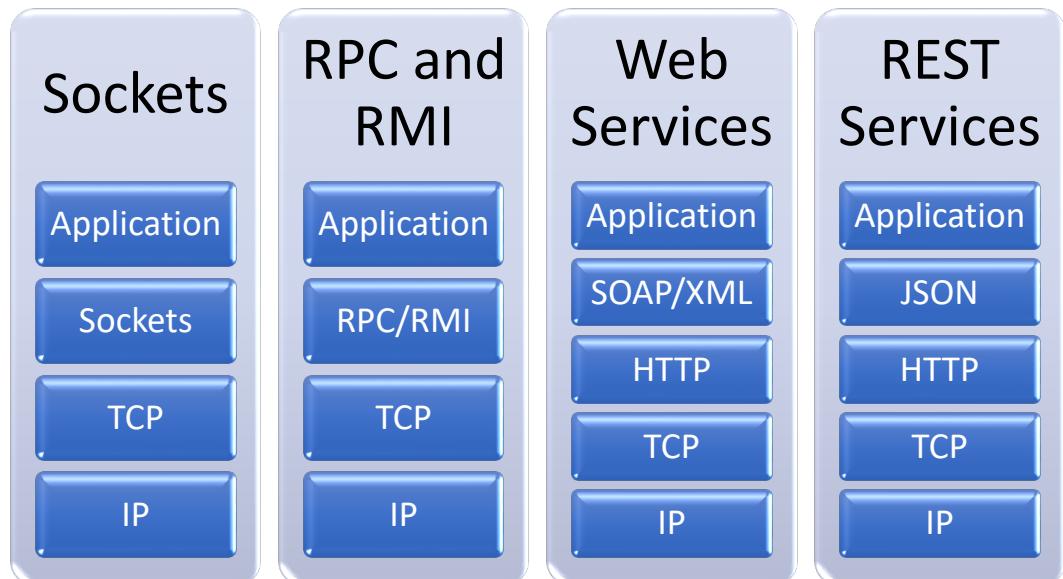
Sergio.guerreiro@tecnico.ulisboa.pt

Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

Arriving to Technological Services

- Evolution of protocol stacks for Request-Response integration
- Services can be consumed in a request-reply pattern, usually blocking the client
- Establishes the separation between a client application and the logic layer exposed as a set of services

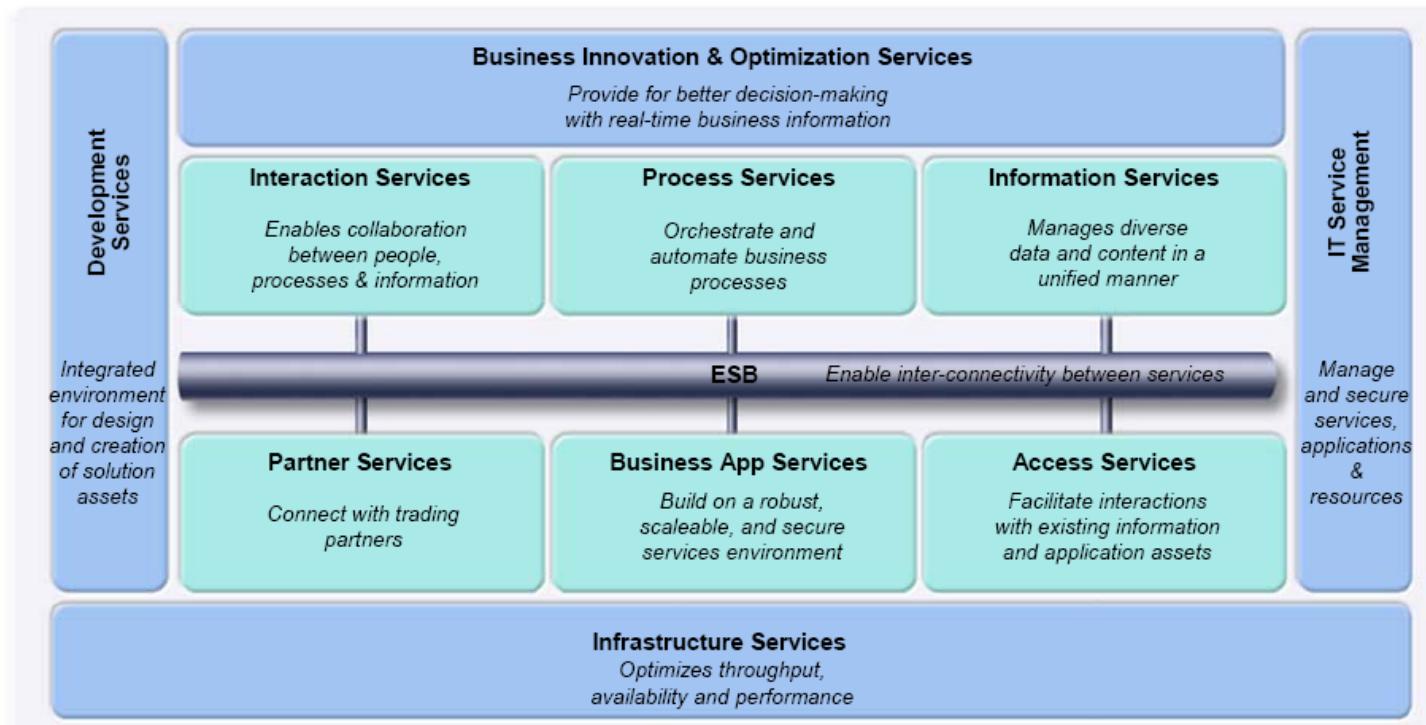


Service Oriented Architecture (SOA)

The classical approach towards services:

- *What?* is a design approach where **multiple services collaborate** to provide some end set of capabilities. A service here typically means a completely separate operating system process. Communication between these services occurs via **calls across a network** rather than method calls within a process boundary.
- *Why?* SOA emerged as an approach to combat the challenges of the large monolithic applications. It is an approach that aims to promote the reusability of software; two or more end-user applications, for example, could both use the same services.
- *Goal to promote reusability:* It aims to make it **easier to maintain or rewrite software**, as theoretically we can replace one service with another without anyone knowing, as long as the semantics of the service don't change too much.

Enterprise Application Integration (EAI)



SOA specifications

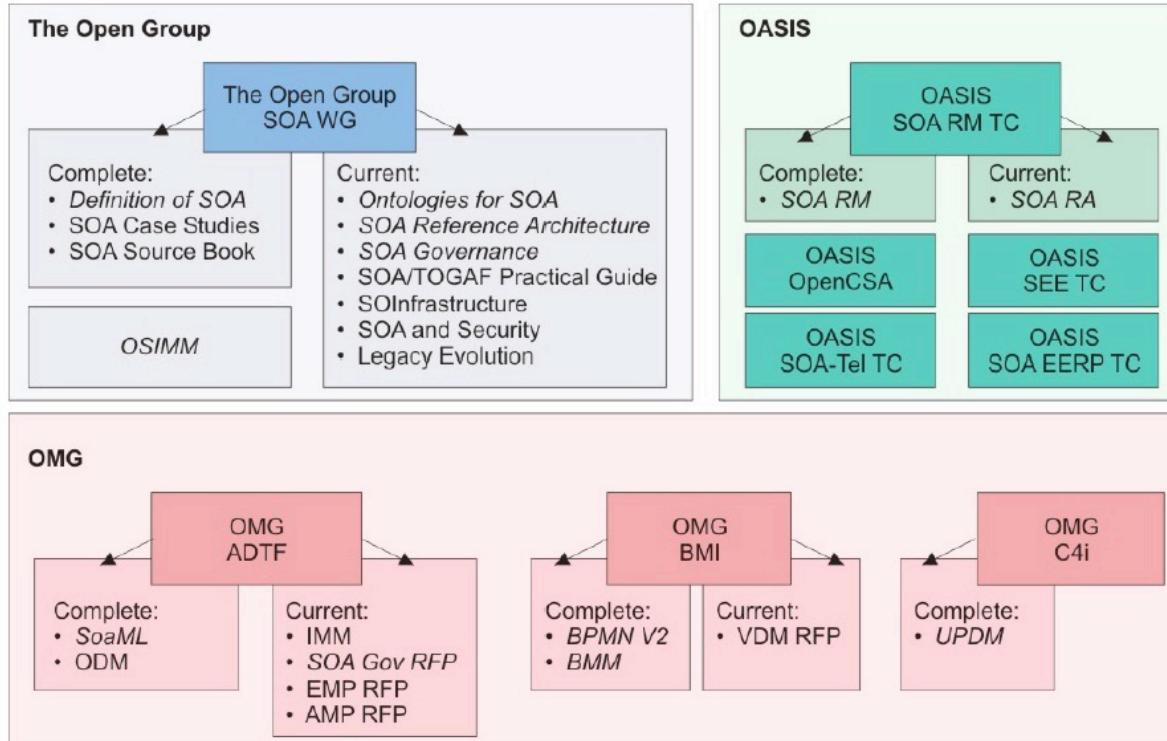


Figure 1: Specifications of SOA Open Standards Working/Work Groups, Technical Committees, and Special Interest Groups

Some SOA pitfalls

- Communication protocols incompatibilities (e.g., SOAP)
- Lack of guidance about service granularity
- Wrong guidance on picking places to split a system
- Many specifications

Back to ArchiMate definitions

- A **business service** represents explicitly defined behavior that a business role, business actor, or business collaboration exposes to its environment.
- An **application service** represents an explicitly defined exposed application behavior.
- A **technology service** represents an explicitly defined exposed technology behavior.



An application service provides a usage contract composed of:

- Explanation of the functionality provided
- The location where the API can be accessed, e.g., HTTP URL to specify the location
- The input and output parameters for the API, such as parameter names, message format, and data types
- The service-level agreement (SLA) that the API provider adheres to such as response time, throughput, availability
- The technical requirements about the rate limits that control the number of requests that an app or user can make within a given period
- Any legal or business constraints on using the API. This can include commercial licensing terms, branding requirements, fees and payments for use, and so on
- Documentation to aid the understanding of the API

Microservices, def. 1

- A microservice is a tiny and independent software process that runs on its own deployment schedule and can be updated independently.

Davis, A. (2021). Bootstrapping Microservices with Docker, Kubernetes, and Terraform: A project-based guide. Simon and Schuster.

- A microservice is an independently deployable component of bounded scope that supports interoperability through message-based communication. Microservice architecture is a style of engineering highly automated, evolvable software systems made up of capability-aligned microservices.

Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2016). Microservice architecture: aligning principles, practices, and culture. "O'Reilly Media, Inc.".

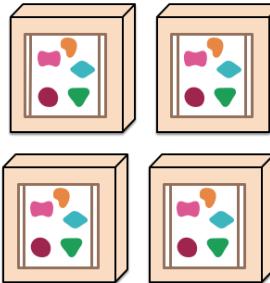
Microservices, def.2

An approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms. [...] built around business capabilities and independently deployable by fully automated deployment machinery

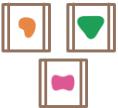
A monolithic application puts all its functionality into a single process...



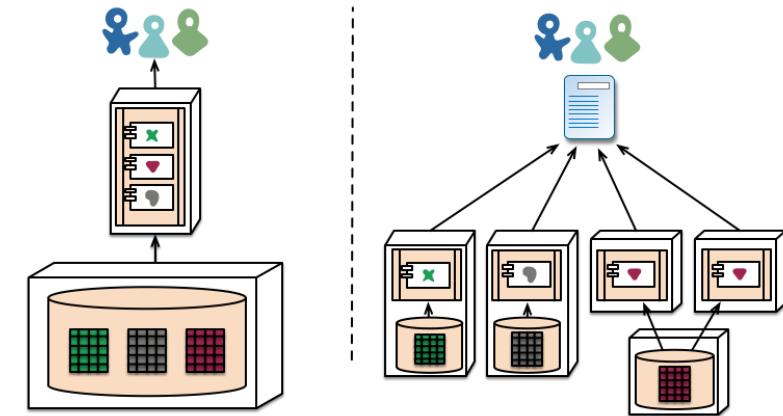
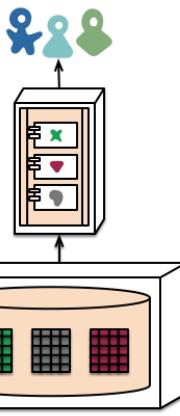
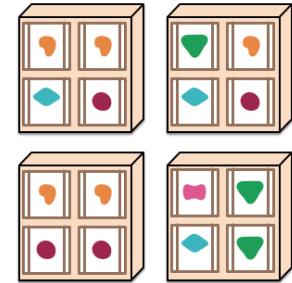
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



<https://martinfowler.com/articles/microservices.html>

Microservices, def.3

- Microservices are independently releasable services that are modelled around a business domain. A service encapsulates functionality and makes it accessible to other services via networks—you construct a more complex system from these building blocks. One microservice might represent inventory, another order management, and yet another shipping, but together they might constitute an entire ecommerce system. Microservices are an architecture choice that is focused on giving you many options for solving the problems you might face.

Newman, S. (2021). Building microservices.
" O'Reilly Media, Inc.".

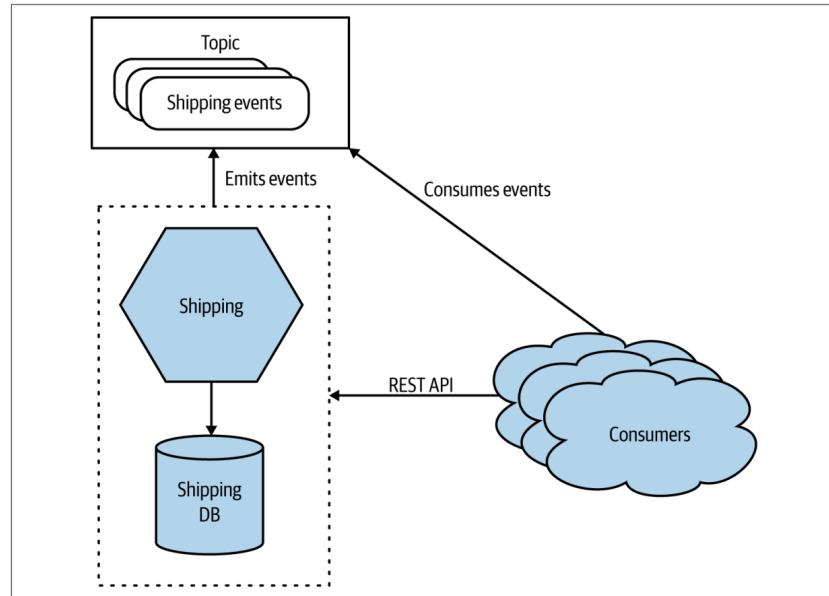


Figure 1-1. A microservice exposing its functionality over a REST API and a topic

Key concepts of microservices to retain

- Run independently
- Independent Deployability
- Distributed data management
 - Owning their own state
 - Bindings to other a bounded context
- Modeled around a Business Domain
 - Size is variable, could small or large. Depends on the business context
- Flexibility
- Alignment with the organization architecture
- Command Query Responsibility Segregation (CQRS)
 - Read/Write your own domain data
 - Read-only representation of other domains data
 - Private data representation (might be different format)

Drawbacks:

- Developers must deal with the additional complexity of creating a distributed system.
- Implementing use cases that span multiple services requires careful coordination between the teams
- Developers must implement the interservice communication mechanism

Asynchronous Event-driven Microservices

Primary benefits:

- **Granularity** – services map neatly to bounded contexts and can be easily rewritten when business requirements change
- **Scalability** – individual services can be scaled up and down as needed
- **Technological flexibility** – services use the most appropriate languages and technologies. This also allows for easy prototyping using pioneering technology.
- **Business requirements flexibility** – ownership of granular microservices is easy to reorganize. There are fewer crossteam dependencies compared to large services, and the organization can react more quickly to changes in business requirements that would otherwise be impeded by barriers to data access
- **Loosely coupling** – event-driven microservices are coupled on domain data and not on specific implementation API. Data schemas can be used to greatly improve how data changes are managed.
- **Continuous delivery support** – it's easy to ship a small, modular microservice, and roll it back if needed
- **High testability** – microservices tend to have fewer dependencies than large monoliths, making it easier to mock out the required testing endpoints and ensure proper code coverage

Types of Coupling

- **Implementation** coupling

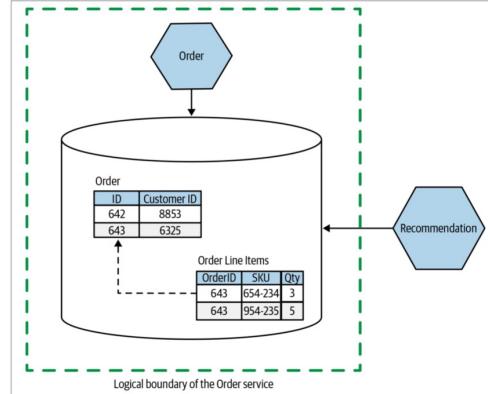


Figure 1-9. The Recommendation service directly accesses the data stored in the Order service

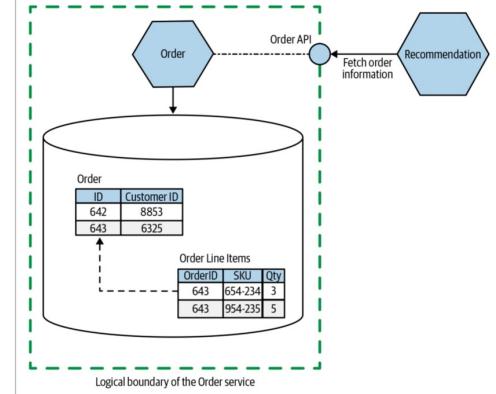


Figure 1-10. The Recommendation service now accesses order information via an API, hiding internal implementation detail

- **Temporal** coupling

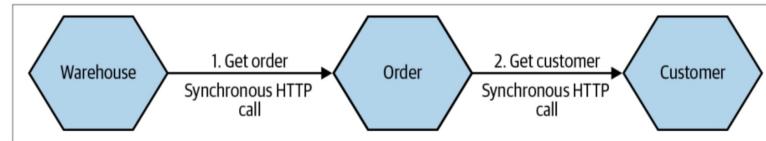


Figure 1-12. Three services making use of synchronous calls to perform an operation can be said to be temporally coupled

- **Deployment** coupling
- **Domain** coupling - the desired coupling for microservices

Enterprise Integration

The *Reactive systems*

Reactive Manifesto classifies reactive systems as being:

- **Responsive** - focus on providing rapid and consistent response times, establishing reliable upper bounds so they deliver a consistent quality of service.
- **Resilient** - The system stays responsive in the face of failure. This applies not only to highly-available, mission critical systems — any system that is not resilient will be unresponsive after a failure. Resilience is achieved by replication, containment, isolation and delegation. Failures are contained within each component, isolating components from each other and thereby ensuring that parts of the system can fail and recover without compromising the system as a whole
- **Elastic** - react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs. This implies designs that have no contention points or central bottlenecks, resulting in the ability to shard or replicate components and distribute inputs among them
- **Message driven** - rely on asynchronous message-passing to establish a boundary between components that ensures loose coupling, isolation, location transparency, and provides the means to delegate errors as messages. Employing explicit message-passing enables load management, elasticity, and flow control by shaping and monitoring the message queues in the system and applying back-pressure when necessary

Reactive 101

- **Responsive** – able to handle requests in a timely fashion
- **Resilient** – able to manage failures gracefully
- **Elastic** – able to scale up and down according to the load and resources
- **Message driven** – using asynchronous message-based communication among the components forming the system

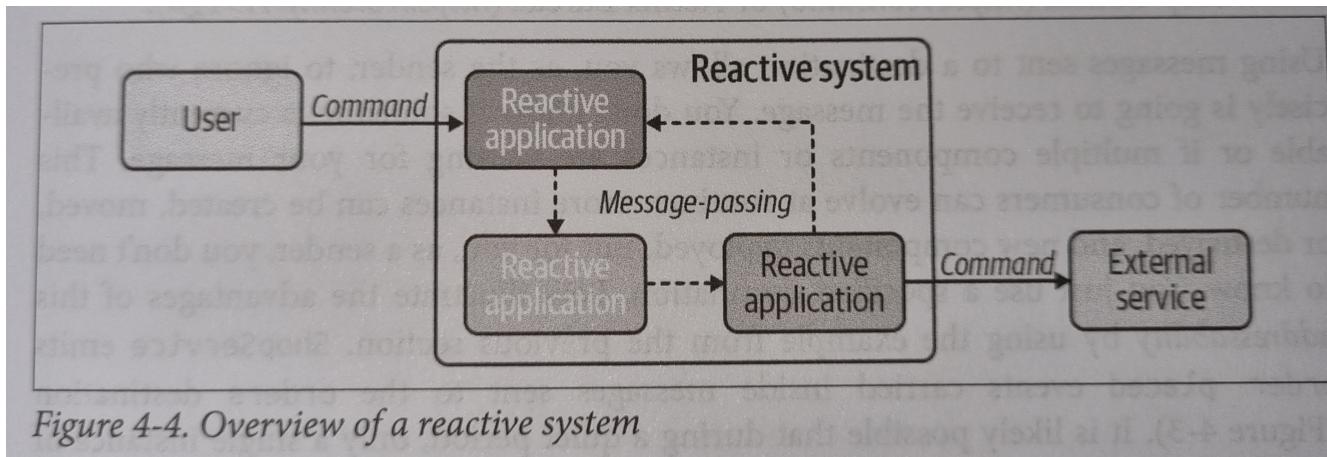
Reactive 101

- **Commands** are actions that a user wishes to perform. The action has not yet happened. It may happen in the future, or not; it may complete successfully or fail. In general, commands are sent to a specific recipient, and a result is sent back to the client
- **Events** are actions that have successfully completed. Represents a fact, something that happened: a keystroke, a failure, an order...
- Events are **immutable**, you cannot change the past. To **refute** a previously sent fact, you need to **fire another event** invalidating the fact
- A **message** is a self-contained data structure describing the event and any relevant details about the event, such as who emitted it, at what time it was emitted, and potentially its unique ID

Overview of a reactive system

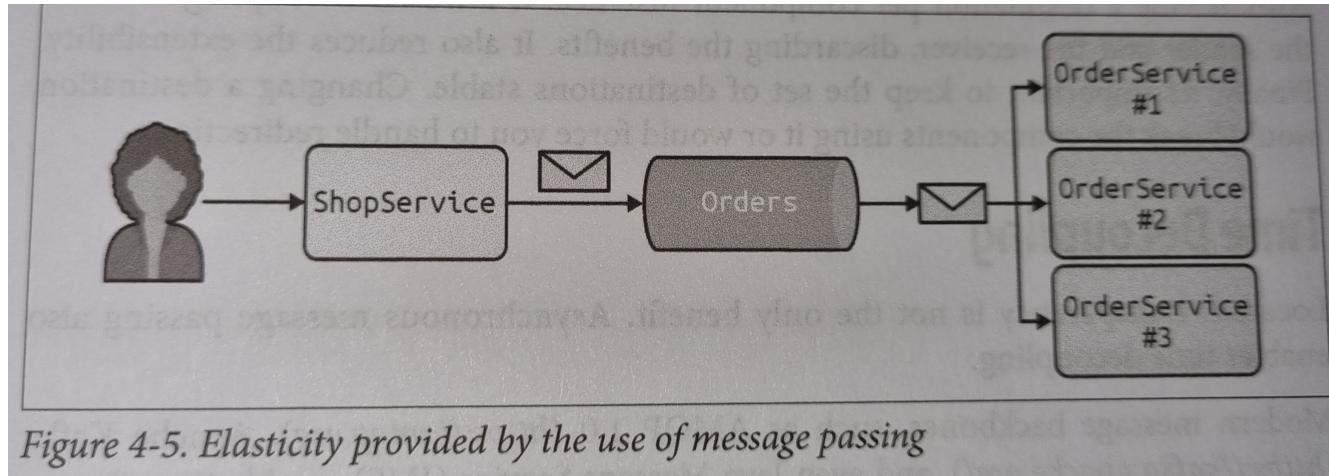
Commands and Messages are the basis of most of interactions. This pattern handles real-world asynchronicity, and binds together services without relying on strong coupling.

At the edge of the system, this approach uses commands most of the time, often relying on HTTP.



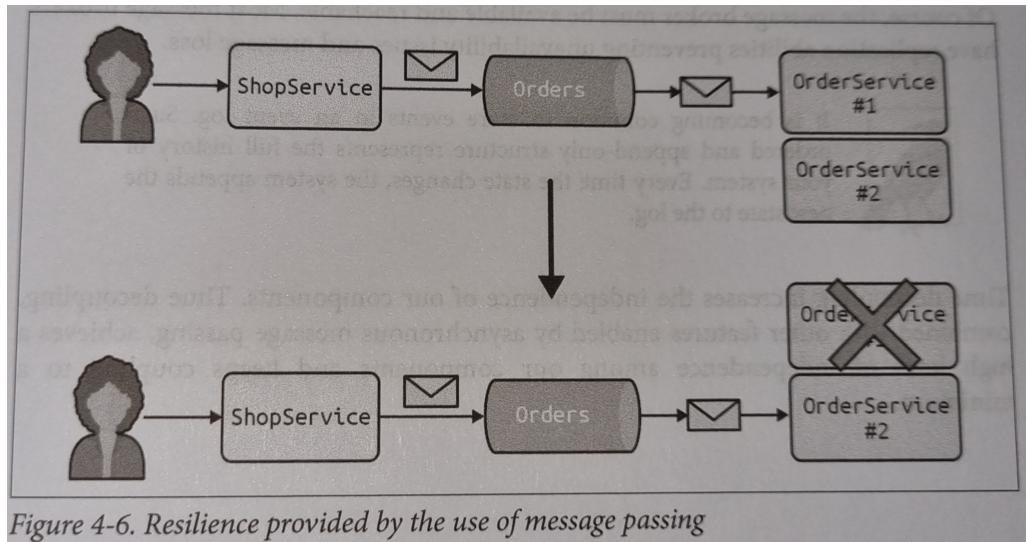
Overview of a reactive system: Elasticity

Scale-from-zero ability, starting OrderService instances to share the systems' load



Overview of a reactive system: Resilience

Message passing enables replication and avoids service disruption



Overview of a reactive system: Time decoupling

- Asynchronous message passing also enables time decoupling.
- Events are not lost if there are no consumers. The events are stored and delivered later.
- Time decoupling increases the independence of components, and keeps coupling to a minimum

The role of Nonblocking Input/Output

- **Blocking network I/O** – synchronous communication where a client and the server connect before interaction starts. Communication is blocked until the operation completes.
- **Multithread blocking network I/O** – Execute concurrent requests having multiple threads. Resources expended waiting for the clients requests. Concurrency limited by the number of threads available.
- **Nonblocking network I/O** – the system enqueues I/O operations and returns immediately, so the caller is not blocked. When the response comes back, the system stores the result in a structure. When the caller needs the result, it interrogates the system to see whether the operation completed.
 - *Continuation-passing style (CPS)* – style of programming in which control is passed explicitly in the form of a continuation (*usually a callback*)

Nonblocking I/O

- Give the possibility to handle multiple concurrent requests or messages with a single thread
- The **reactor pattern** allows associating I/O events with event handlers
- Invokes the event handlers when the expected event is received
- Avoiding the creation of a thread for each message, request and connection

it's a thread iterating over the set of channels

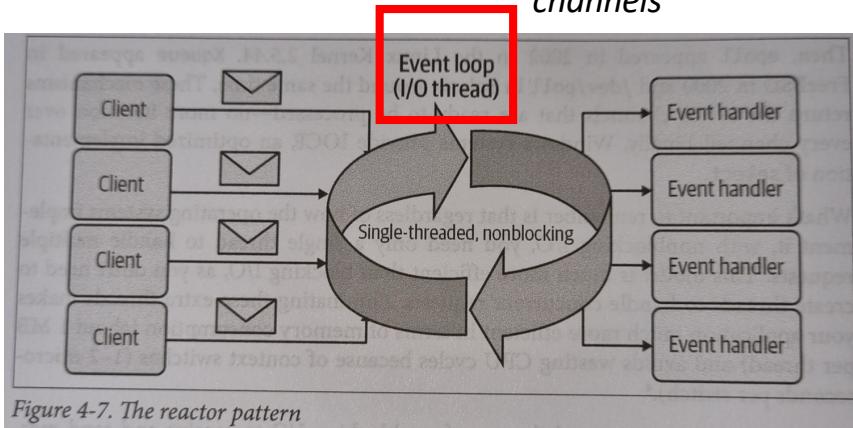


Figure 4-7. The reactor pattern

Nonblocking & Blocking I/O

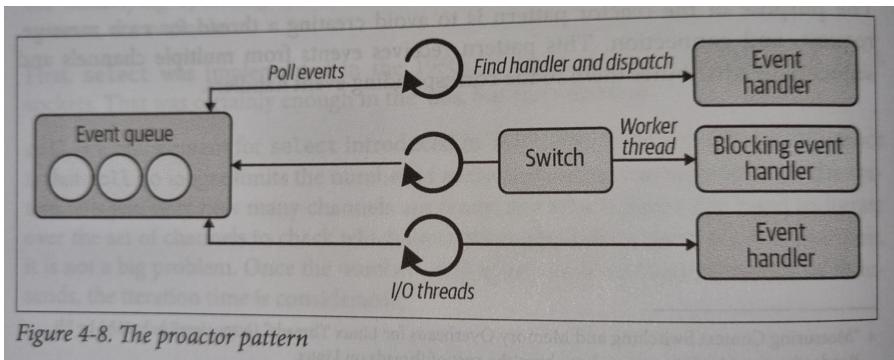


Figure 4-8. The proactor pattern

- The **Proactor pattern** can be seen as an asynchronous version of the reactor
- Useful when long-running event handlers invoke a continuation when they complete
- Such mechanisms allow mixing nonblocking and blocking I/O

The common architecture

- Bottom layer, handles client connections, outbound requests, and response writing.
- Middle layer, provides easier and high-level APIs such as HTTP requests, responses, Kafka messages
- Top layer, the code developed by you, that is just a collection of event handlers. Uses the features provided by the reactive framework to interact with other services or middleware.
 - **However, your code cannot block the event loop thread, otherwise the architecture will be blocked!**
 - **Design & develop non-blocking code, always!**

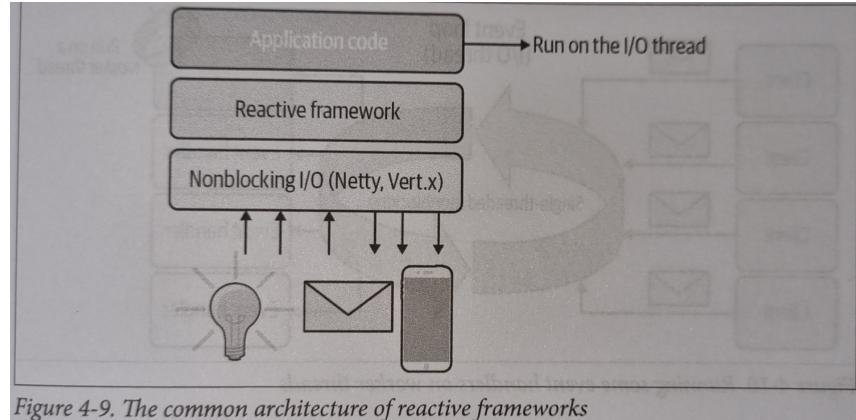
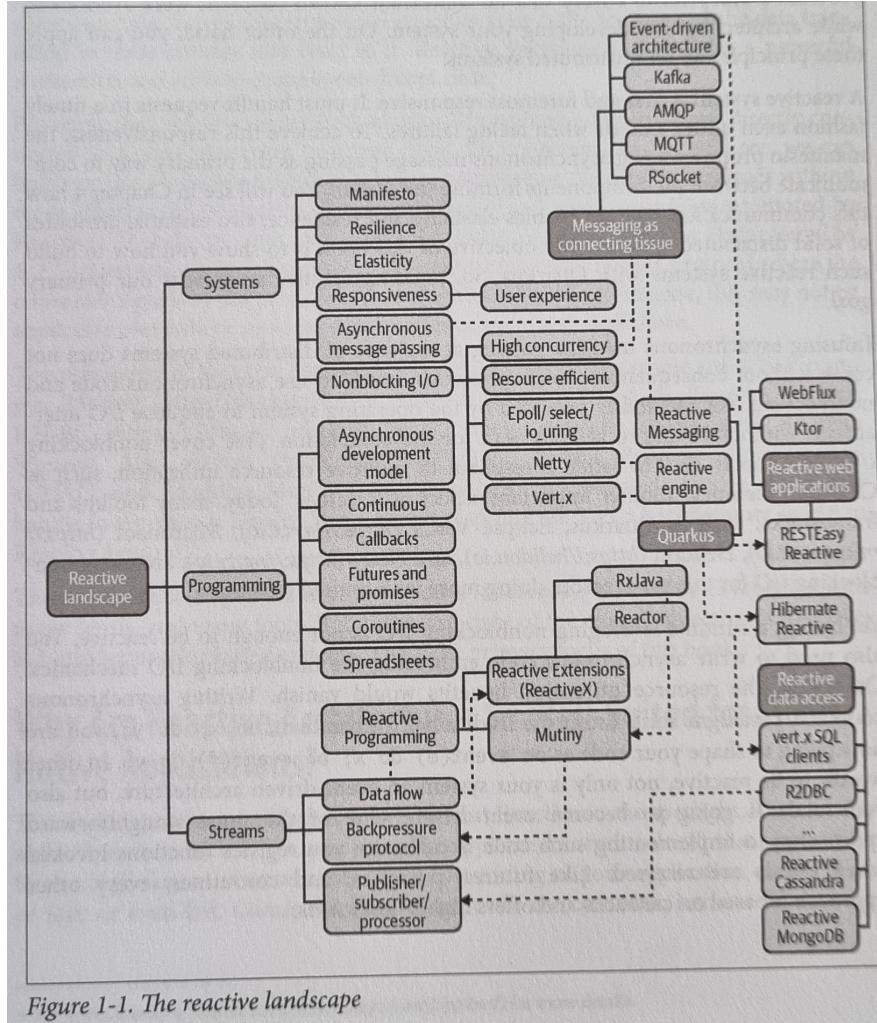


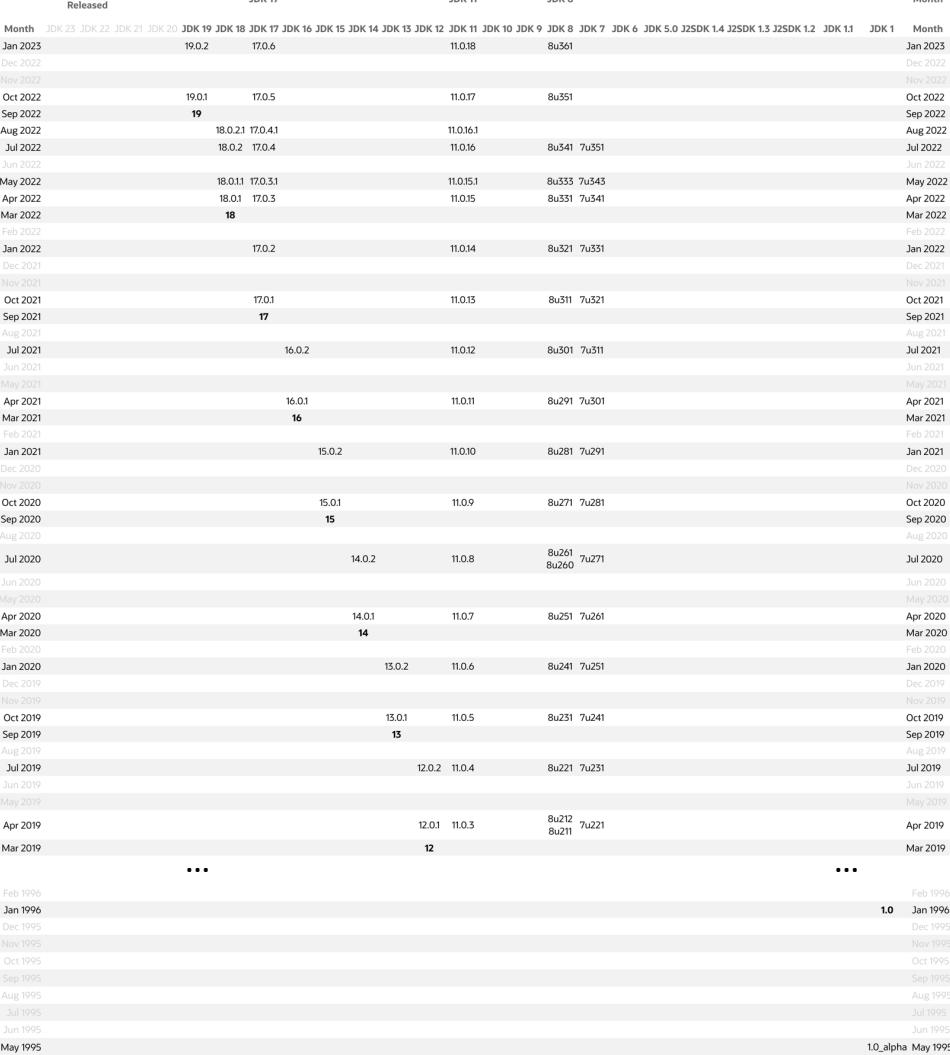
Figure 4-9. The common architecture of reactive frameworks

The concepts of the Reactive Landscape



Enterprise Integration

Quarkus Framework



<https://www.java.com/releases/fullmatrix/>

Java history

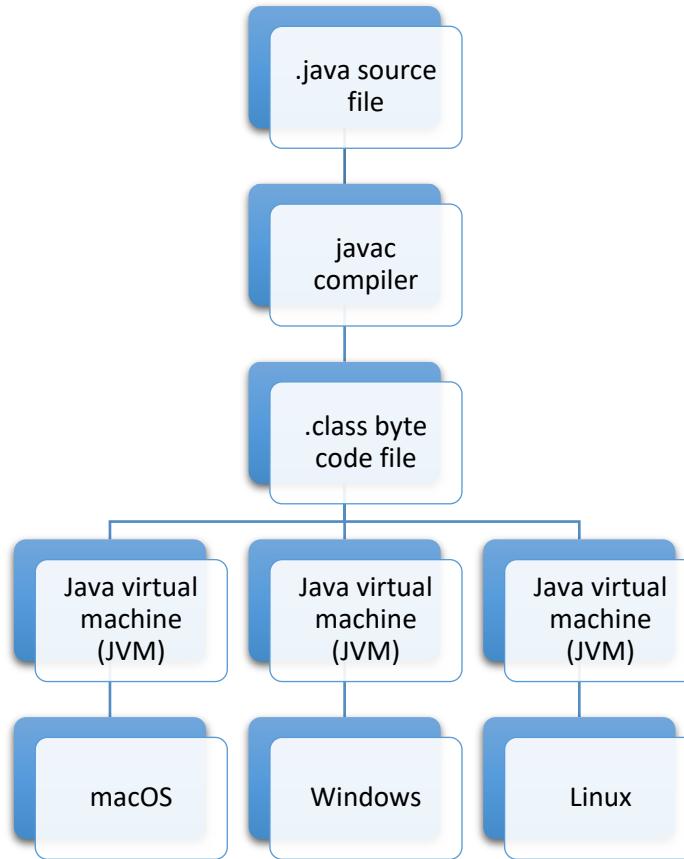
JAVA roadmap

Planned ⓘ

Month	JDK 23	JDK 22	JDK 21	JDK 20	JDK 19	JDK 18	JDK 17	JDK 16	JDK 15	JDK 14	JDK 13	JDK 12	JDK 11	JDK 10	JDK 9	JDK 8	JDK 7	JDK 6	JDK 5.0	J2SDK 1.4	J2SDK 1.3	J2SDK 1.2	JDK 1.1	JDK 1	Month
Jan 2025	23.0.2		21.0.6			17.0.14							11.0.26			8u441									Jan 2025
Dec 2024																									Dec 2024
Nov 2024																									Nov 2024
Oct 2024	23.0.1		21.0.5			17.0.13							11.0.25			8u431									Oct 2024
Sep 2024	23																								Sep 2024
Aug 2024																									Aug 2024
Jul 2024		22.0.2	21.0.4			17.0.12							11.0.24			8u421									Jul 2024
Jun 2024																									Jun 2024
May 2024																									May 2024
Apr 2024		22.0.1	21.0.3			17.0.11							11.0.23			8u411									Apr 2024
Mar 2024		22																							Mar 2024
Feb 2024																									Feb 2024
Jan 2024			21.0.2			17.0.10							11.0.22			8u401									Jan 2024
Dec 2023																									Dec 2023
Nov 2023																									Nov 2023
Oct 2023			21.0.1			17.0.9							11.0.21			8u391									Oct 2023
Sep 2023			21																						Sep 2023
Aug 2023																									Aug 2023
Jul 2023				20.0.2		17.0.8							11.0.20			8u381									Jul 2023
Jun 2023																									Jun 2023
May 2023																									May 2023
Apr 2023					20.0.1	17.0.7							11.0.19			8u371									Apr 2023
Mar 2023					20																				Mar 2023

JAVA features

- Object oriented
- Simple
- Robust
- Multithreaded
- Distributed
- Dynamic
- Platform independent
- Architecture-neutral
- Portable
- Interpreted
- Garbage collector



JAVA Enterprise Edition (EE)



- Java EE, Java2EE, J2EE, Jakarta EE are evolving names of a set of enterprise specifications that extends JAVA EE

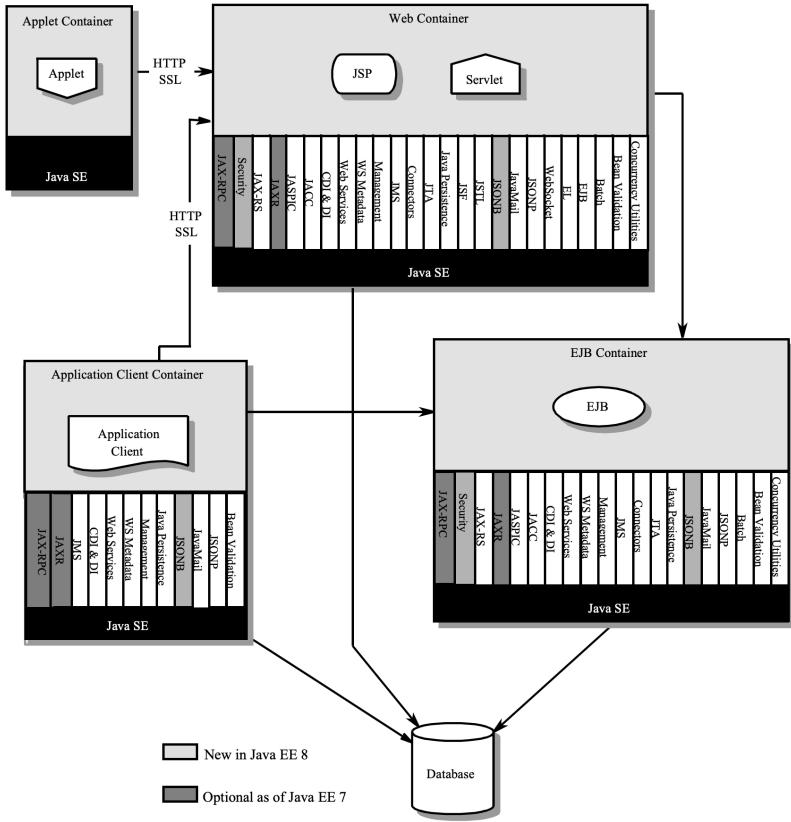
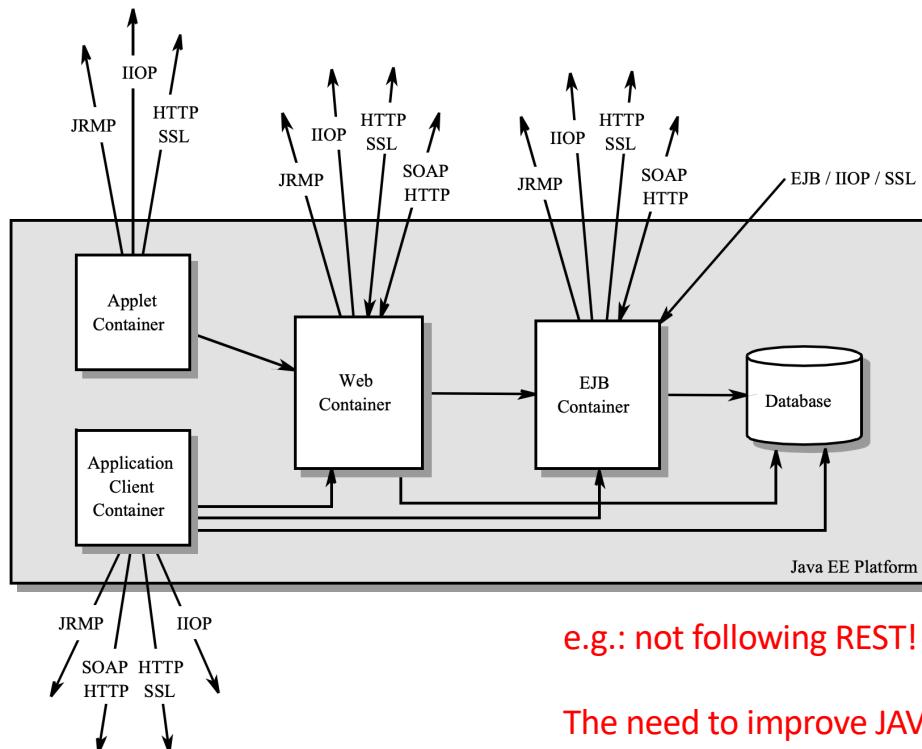


Figure EE.2-1 Java EE Architecture Diagram

JAVA EE Interoperability



Quarkus framework on JAVA

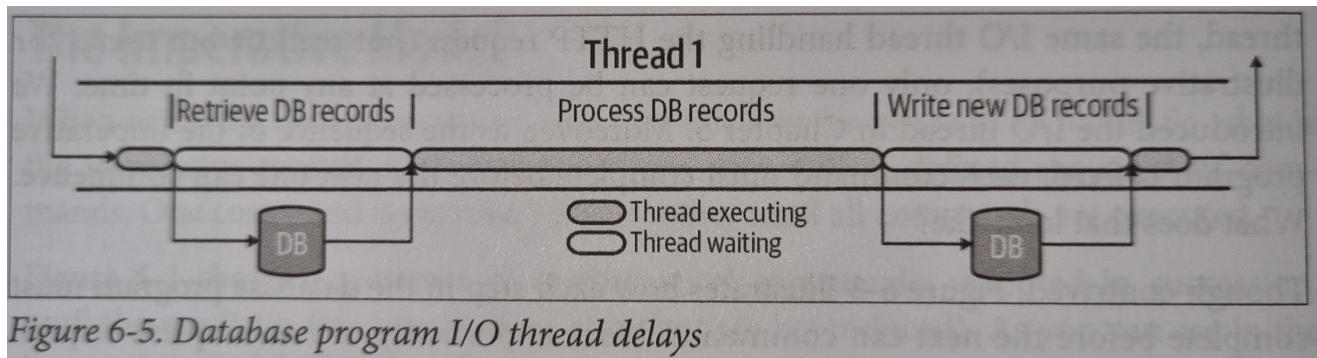
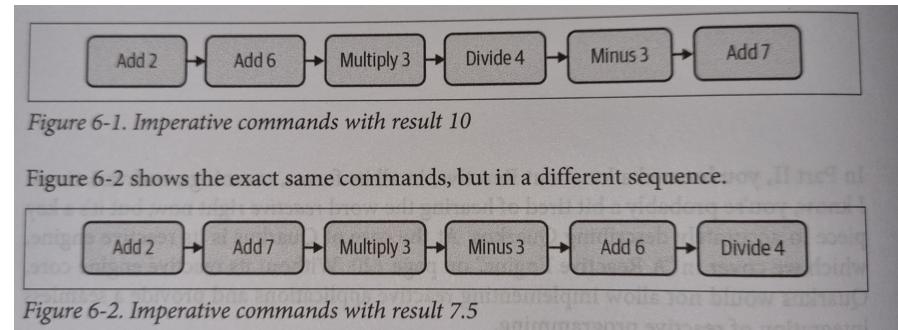
Two distinct models to design
your code...

The Imperative Model

The order of the commands cannot be changed or the result will be different.

Sometimes, is the only possible way!

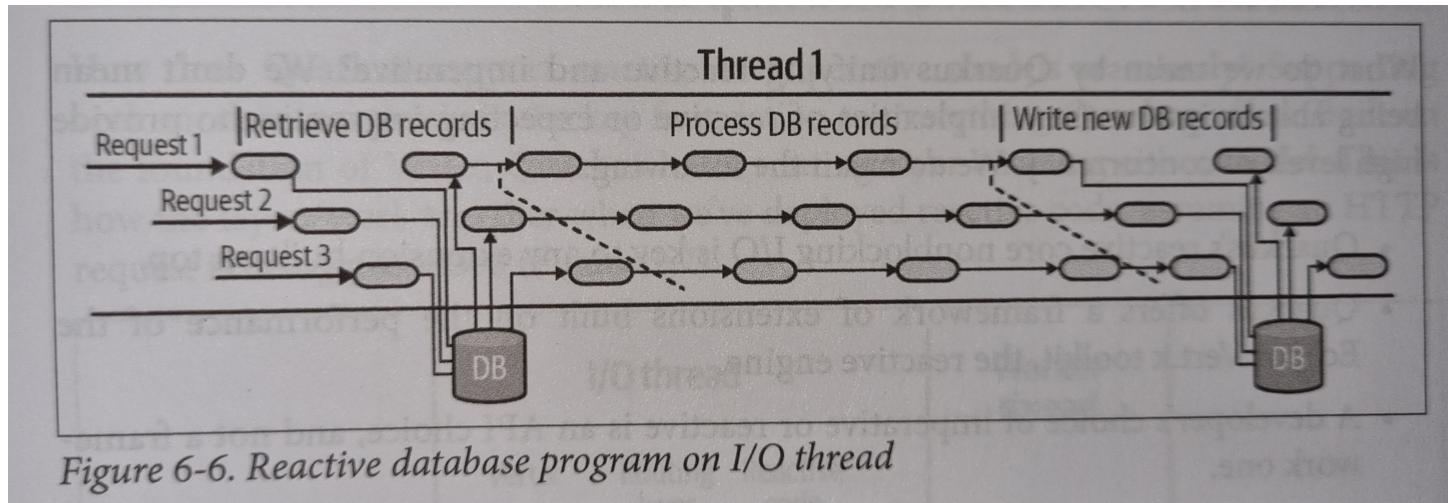
However, in between sending a request to the database and receiving the response, what is the I/O thread doing?



Many inefficiencies

The Reactive Model

- Instead of an I/O thread waiting, it begins processing another incoming request. It continues to do so until it's been notified that a database response is ready for processing.
- How? A continuation (*a callback*) is provided to process the database response



The Quarkus supported Models

- A **developer's choice** of imperative or reactive is an API choice, and not a framework one
- Quarkus reactive model is always non-blocking, relying on **Eclipse Vert.x**
- Imperative model requires the execution by a **worker thread** and not I/O thread. Offload from Quarkus is done using **context switch**. **However is cost time and resources.**

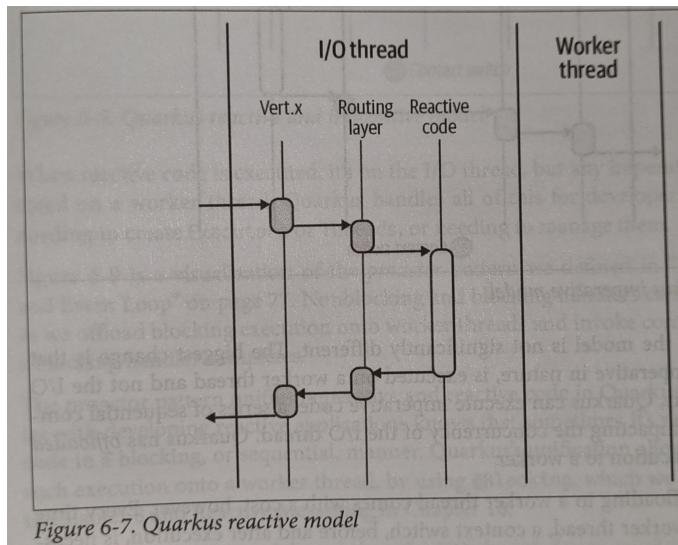


Figure 6-7. Quarkus reactive model

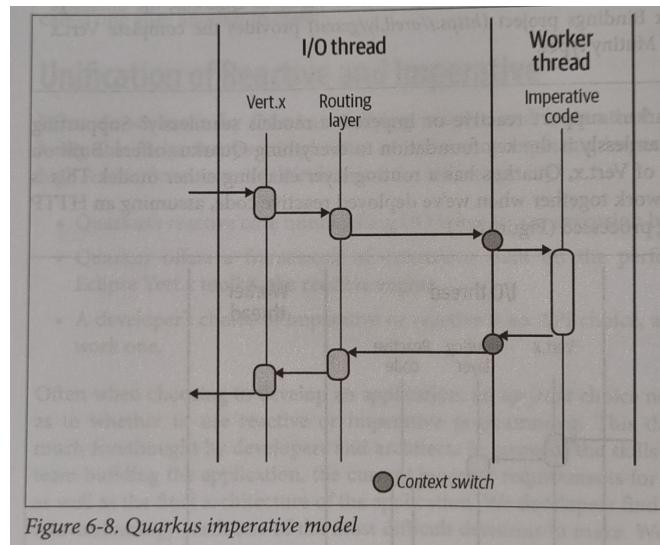


Figure 6-8. Quarkus imperative model

Unification of Reactive and Imperative

- Yet, Third option!
 - Non-blocking and blocking handlers can coexist, as long as we offload blocking execution onto worker threads and invoke continuations when a blocking handler completes.
 - Using I/O thread for as much work as possible
- How?
 - Using the `@Blocking` and `@NonBlocking` annotations

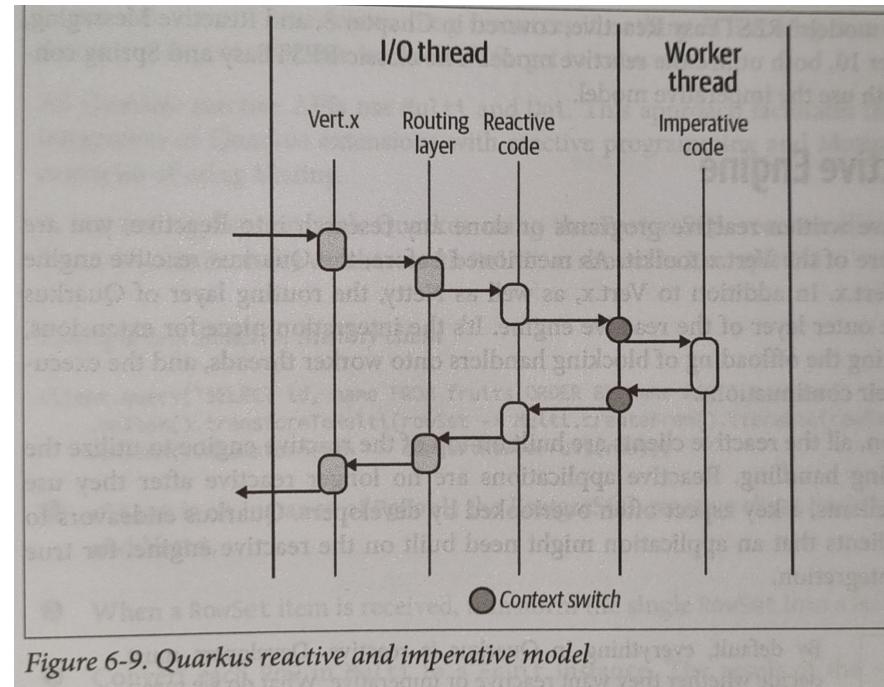
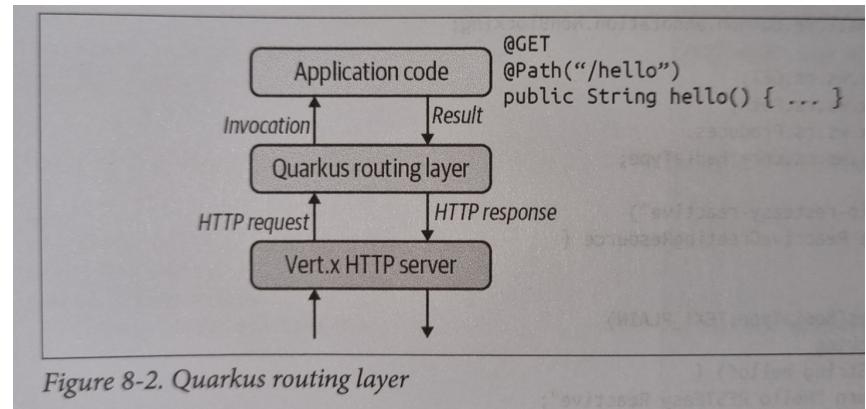


Figure 6-9. Quarkus reactive and imperative model

RESTEasy

- With RESTEasy Reactive the annotations `@Blocking` and `@NonBlocking` can be used to indicate on which threads the request should be handled
- By default, `@NonBlocking` uses an I/O Thread**
- A method returning Uni or Multi is executed on an I/O thread except if annotated with `@Blocking` that uses a worker Thread**
- Methods returning any other object is executed on a worker thread, except if the `@NonBlocking` annotation is used**



RESTEasy

	Annotation required to the method returning Uni or Multi	Annotation required to the method returning any other datatype
I/O Thread	@NonBlocking (it's by default)	@NonBlocking
Worker Thread	@Blocking	Default: @Blocking

Mutiny = *Multi and Uni*

- SmallRye Mutiny is the reactive programming library of Quarkus.
- Mutiny is built around three key aspects:
 - Event-driven: listening to events from stream and handling them appropriately
 - Easily navigable API: Navigating the API is driven by an event type and the available options for that event
 - Only two types: **Uni** and **Multi** can handle any desired asynchronous actions

	Events	Use cases	Implement Reactive Streams
Uni	Item and failure	Remote invocation, asynchronous computation returning a single result	No
Multi	Item, failure, completion	Data streams, potentially unbounded (emitting an infinite number of streams)	Yes

Mutiny = Multi and Uni

```
@GET
```

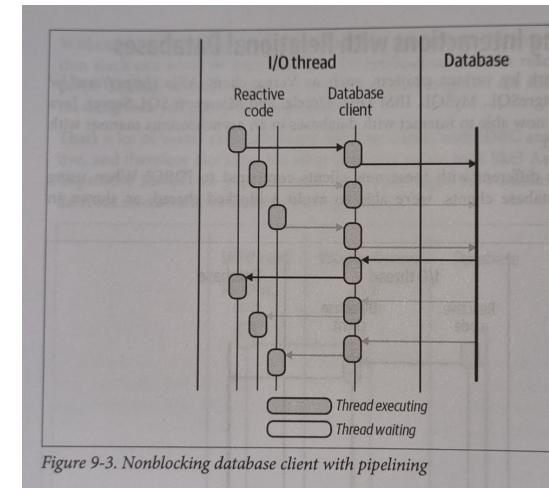
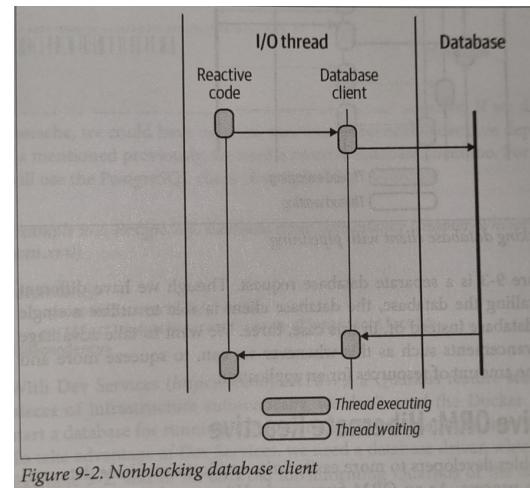
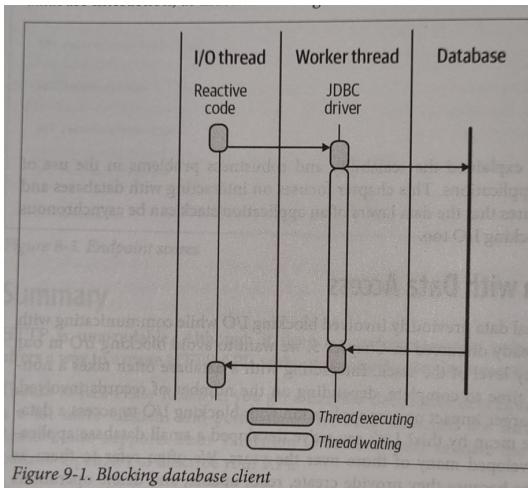
```
public Multi<Fruit> get() {  
    return Fruit.findAll(client);  
}
```

```
@POST
```

```
public Uni<Response> create(Fruit fruit) {  
    return fruit.save(client)  
        .onItem().transform(id -> URI.create("/fruits/" + id))  
        .onItem().transform(uri -> Response.created(uri).build());  
}
```

Reactive Object-relational mapping (ORM)

- Three distinct models available:
 - Blocking database client
 - Non-blocking database client
 - Non-blocking database with pipelining – database shared connection (*for compatible databases*)



Hibernate Reactive

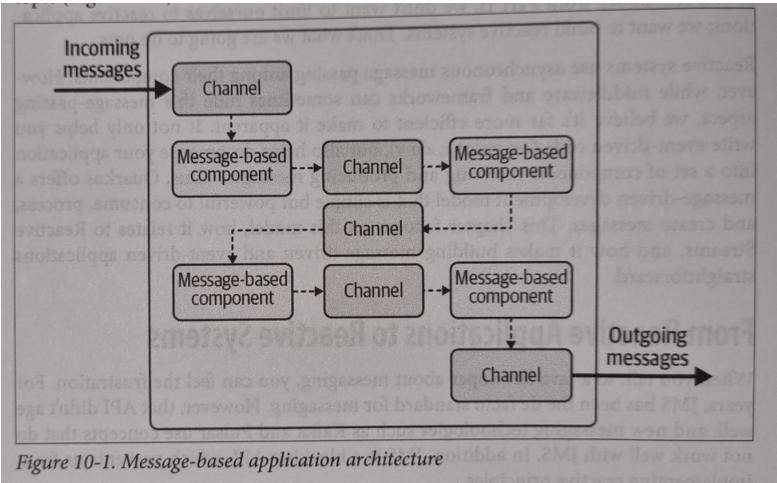
- Hibernate ORM enables developers to more easily write applications whose data outlives the application process. As an ORM framework, Hibernate is concerned with data persistence as it applies to relational databases. Hibernate provides both imperative and reactive APIs
- These APIs support two facets:
 - non-blocking database clients, and
 - reactive programming as a means of interacting with relational databases.

```
mvn io.quarkus.platform:quarkus-maven-plugin:2.7.5.Final:create \
-DprojectId=org.acme \
-DprojectArtifactId=reactive-mysql-transactioncontrol-client-quickstart \
-Dextensions="resteasy-jackson,quarkus-reactive-mysql-client,resteasy-mutiny,quarkus-smallrye-openapi" \
-DnoCode

mvn io.quarkus.platform:quarkus-maven-plugin:2.7.5.Final:create \
-DprojectId=org.acme \
-DprojectArtifactId=reactive-pg-client-quickstart \
-Dextensions="resteasy-jackson,reactive-pg-client,resteasy-mutiny,quarkus-smallrye-openapi" \
-DnoCode
```

Reactive Messaging: Quarkus with Kafka

- Quarkus offers a message-driven development model that is simple but powerful to consume, process, and create messages
- Old JAVA JMS is not fully compatible with Kafka and is a blocking API which do not allow the implementation of reactive principles
- Reactive Messaging can send, consume, and process messages in a protocol-agnostic way
- Messages transit on **channels**



```
@Channel("my-channel")
MutinyEmitter<Person> personEmitter;

public Uni<Void> send(Person p) {
    return personEmitter.send(p);
}
```

Producing messages

```
@Incoming("words-in")
@Outgoing("words-out")
public Message<String> todb(Message<String> message)
{
    return message.withPayload(message.getPayload().toUpperCase());
}
```

Processing messages

Q&A





TÉCNICO LISBOA

Enterprise Integration

Containers & Cloud

Prof. Sérgio Guerreiro

Sergio.guerreiro@tecnico.ulisboa.pt

Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

Cloud services properties

- **Broadband access** - Consume the services from anywhere
- **On-demand self-service** - Consume the services when you want
- **Resource pooling and virtualization** - Pool the infrastructure, virtual platforms and applications
- **Rapid elasticity** - Pooled resources with horizontal scalability
- **Measured service** - Pay only for what you consume when you consume

Cloud services

SaaS	Software As a Service				
PaaS	Platform As a Service				
IaaS	Infrastructure As a Service				
IaC	Infrastructure as Code		Terraform		

IaaS vs. PaaS vs. SaaS

You have more complete control over the configuration of your cloud resources in IaaS than in PaaS and SaaS. PaaS and SaaS virtualize more infrastructure functions and you have fewer components to manage as compared to IaaS.

Consider the following table. If you manage your own IT infrastructure, you must invest in and maintain all the items mentioned in the table. On the other hand, if you switch to a particular cloud computing service, this is how it works:

- Yellow cells indicate what you manage
- Green cells indicate what the cloud service provider manages

	Own IT	IaaS	PaaS	SaaS
Application		Y	Y	Y
Data		Y	Y	G
Runtime or software that runs the application		Y	Y	G
Middleware or software that monitors the application		Y	G	G
Operating systems on which the application runs		Y	G	G
Virtualization technology		Y	G	G
Server machines		Y	G	G
Storage devices		Y	G	G
Network appliances		Y	G	G

<https://aws.amazon.com/what-is/iaas/>

Clouds may be hosted and employed in different styles depending on the use case, respectively the business model of the provider: **Private** cloud; **Community** cloud; **Public** cloud; **Hybrid** cloud; **Special purpose** clouds.

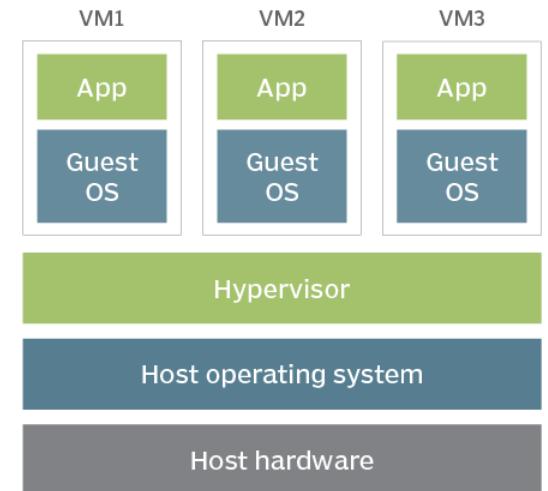
Physical Servers

- Slow-iteration and slow-deployment
- Single tenancy
- Unfriendly for friendly for multi programming languages
- Deploy in weeks
- Typically, alive for years



Virtual Machines (VMs)

- VMs work by operating on top of a hypervisor, which is stacked on top of a host machine
- A VM monitor (VMM) or **hypervisor** intermediates between the host and guest VM. By isolating individual guest VMs from each other, the VMM enables a host to support multiple guests running different OSes.
- Each VM carries their own virtualized hardware stack that comprises network adapters, storage, applications, binaries, libraries and its own CPU
- Advantages:
 - VMs allow to consolidate applications onto a single server.
 - Faster iteration and deployment
 - Multi-tenancy
 - Somewhat friendly for multi programming languages
 - Deploy in minutes
 - Typically, alive for weeks
- Disadvantages: Having multiple VMs with their own OS adds substantial **overheads** in terms of RAM, CPU, I/O and storage



Containers

- A container is a **self-contained** execution environment that shares the kernel of the host system and which is (optionally) isolated from other containers in the system
- One of the major advantages of containers is resource **efficiency**, because you don't need a whole operating system instance for each isolated workload
- When a process is running inside a container, there is only a **little bit of code** that sits inside the kernel managing the container.
- Contrast this with a virtual machine where there would be a second layer running. In a VM, **calls by the process to the hardware or hypervisor** would require bouncing in and out of privileged mode on the processor twice, thereby noticeably slowing down many calls.

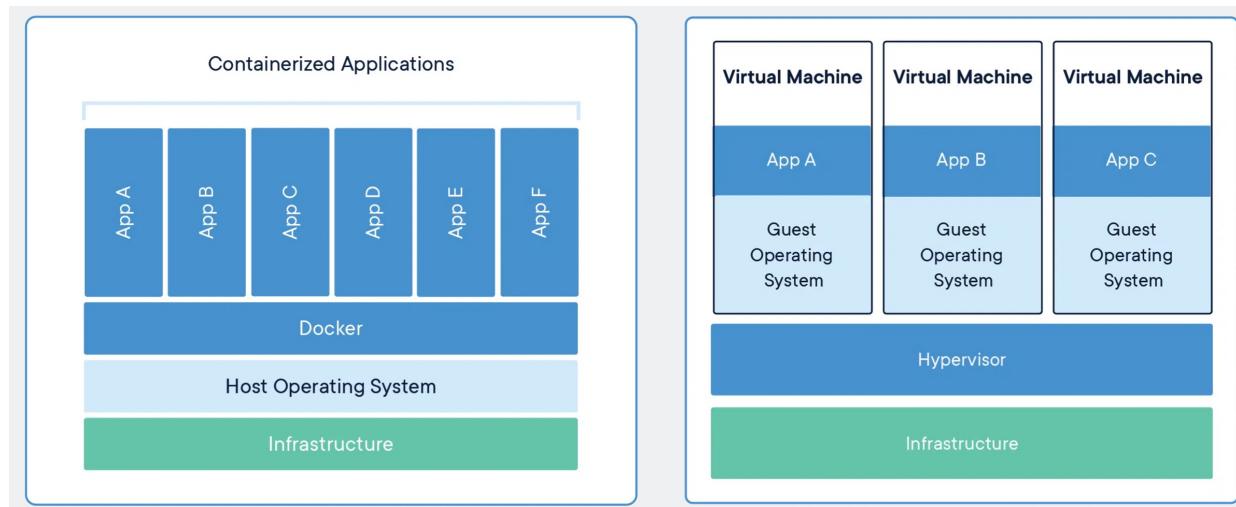
Docker containerization

Containers are an **abstraction at the app layer** that packages code and dependencies together.

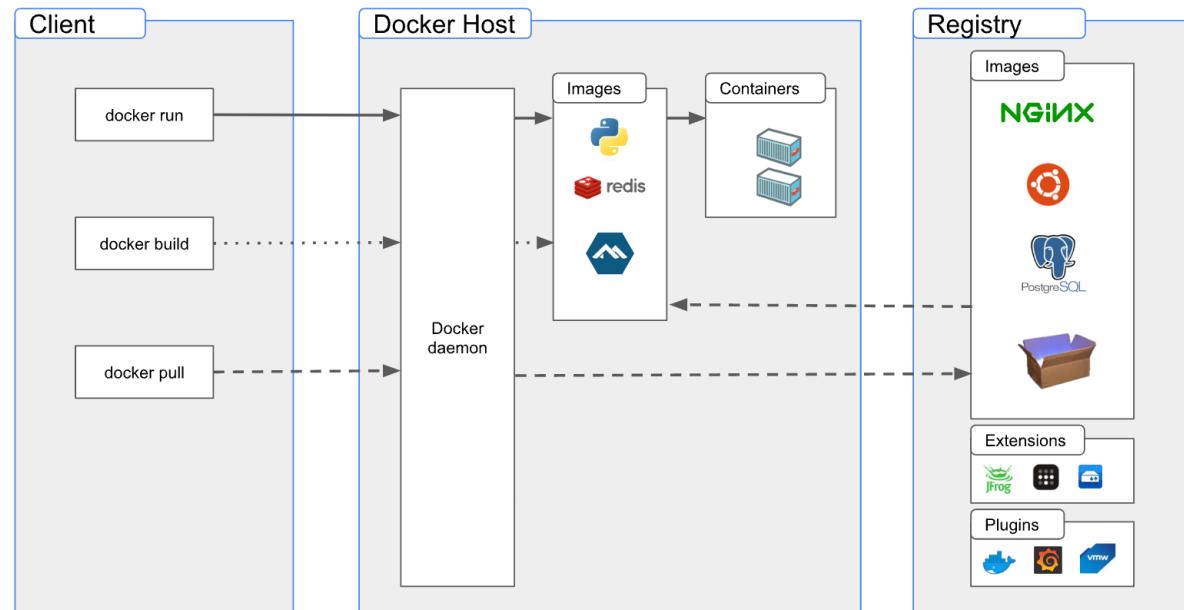
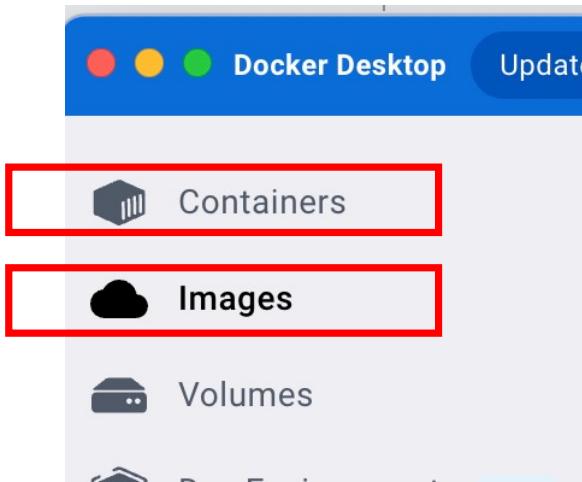
Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

Advantage: more lightweight than VM's

Containers sit on top of a Docker engine.

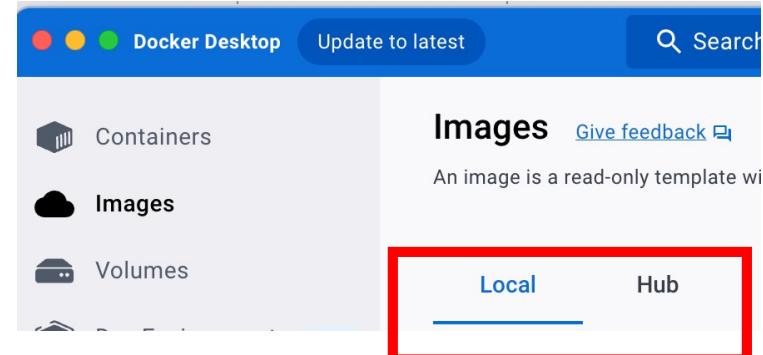


Docker core concepts



Docker image definition

- An *image* is a **read-only template** with instructions for creating a Docker container. Often, an image is *based on* another image, with some additional customization.
 - For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.
- You might create your **own images** or you might only use those created **by others** and published in a registry. To build your own image, you create a *Dockerfile* with a simple syntax for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt. This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.



Docker container *definition*

- A container is a runnable instance of an image. You can **create, start, stop, move, or delete** a container using the Docker API or CLI. You can connect a container to **one or more networks, attach storage** to it, or even **create a new image** based on its current state.
- By default, a container is relatively well isolated from other containers and its host machine. You can control **how isolated** a container's network, storage, or other underlying subsystems are from other containers or from the host machine.
- A container is defined by its image as well as any configuration options you provide to it when you create or start it.
- When a container is removed, any changes to its state that are not stored in persistent storage disappear.

Example docker run command

- The following command runs an ubuntu container, attaches interactively to your local command-line session, and runs /bin/bash.

```
$ docker run -i -t ubuntu /bin/bash
```

Docker Hub

Is a service provided by Docker for finding and sharing container images with others. It's the world's largest repository of container images with an array of content sources including container community developers, open source projects and independent software vendors (ISV) building and distributing their code in containers.

Docker Hub provides the following major features:

[Repositories](#): Push and pull container images.

[Teams & Organizations](#): Manage access to private repositories of container images.

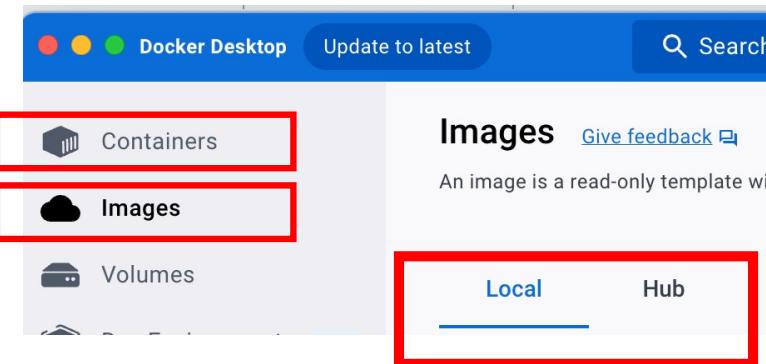
[Docker Official Images](#): Pull and use high-quality container images provided by Docker.

[Docker Verified Publisher Images](#): Pull and use high-quality container images provided by external vendors.

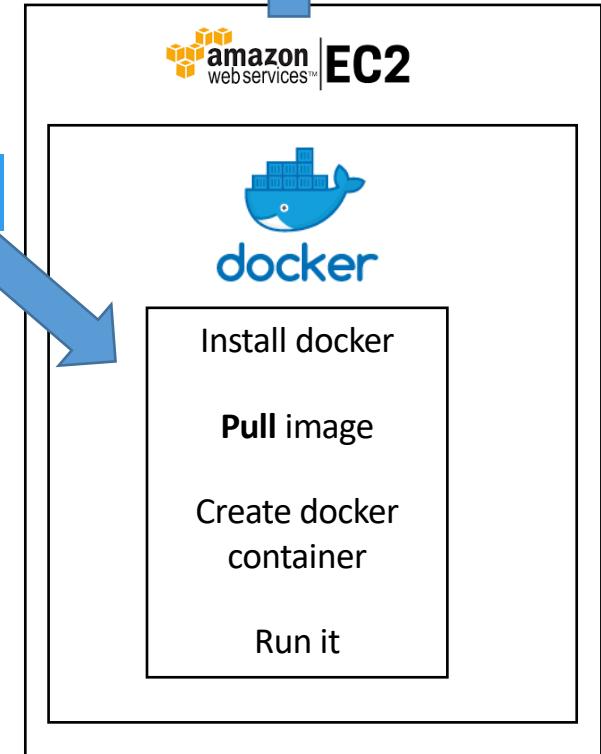
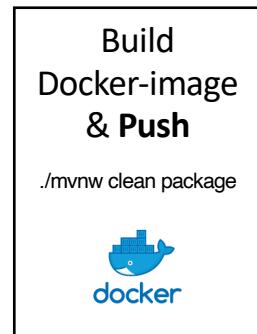
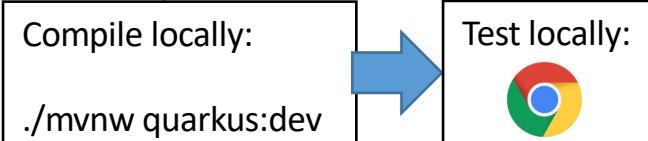
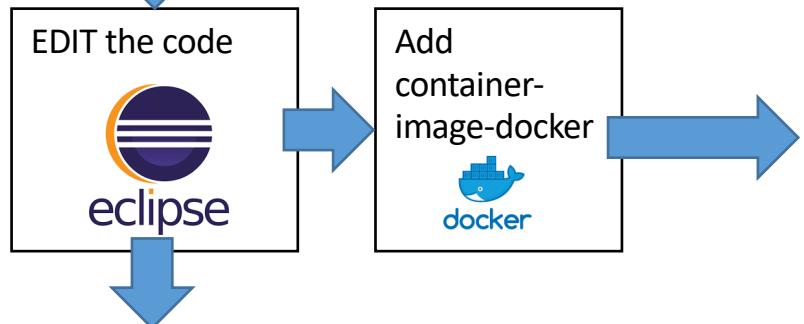
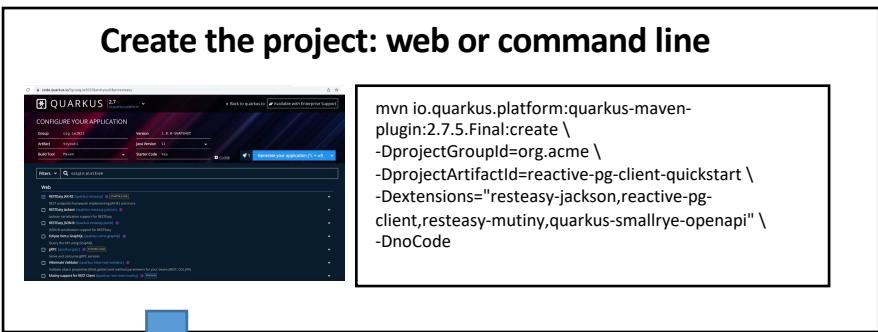
[Docker-Sponsored Open Source Images](#): Pull and use high-quality container images from non-commercial open source projects.

[Builds](#): Automatically build container images from GitHub and Bitbucket and push them to Docker Hub.

[Webhooks](#): Trigger actions after a successful push to a repository to integrate Docker Hub with other services.



Docker instance inside EC2 instance by dockerhub!



Docker compose *definition*

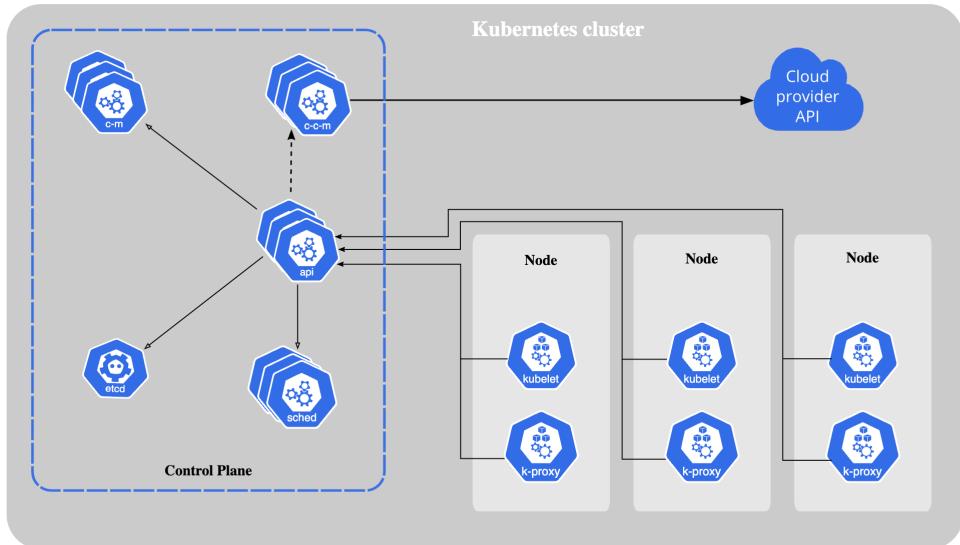
- Is a tool for defining and running **multi-container Docker applications**.
- a YAML file is used to configure the application's services. Then, with a command, you create and start all the services from your configuration.
- Compose works in all environments: production, staging, development, testing, as well as CI workflows. It also has commands for managing the whole lifecycle of applications:
 - Start, stop, and rebuild services
 - View the status of running services
 - Stream the log output of running services
 - Run a one-off command on a service

But, how to scale with more flexibility?

Kubernetes

- Is an open source orchestrator for deploying, scaling, and management of **containerized applications** (*different from terraform which is applicable to any cloud resource*)
- It allows to run Docker containers and workloads and helps tackling some of the operating complexities when moving to scale multiple containers, deployed across multiple servers
- The main benefits are
 - The service delivery velocity founded on
 - **Immutability**: rather than incremental updates and changes, an entirely new, complete image is built, where the update simply replaces the entire image with the newer image in a single operation
 - **Declarative configuration**: describing the state of the desired world, instead of specifying the series of instructions
 - **Online self-healing systems**: e.g., if you assert a desired state of 3 replicas, Kubernetes creates exactly 3 replicas. If manually a fourth is created, Kubernetes destroys it.
 - Scaling (software and teams)
 - Decoupling
 - Clusters
 - Microservices
 - Separation of concerns
 - Abstracting the infrastructure
 - Resource efficiency usage

Kubernetes Architecture



- A **node** is a machine where containers (workloads) are deployed. Every node in the cluster must run a container runtime such as Docker, as well as the below-mentioned components, for communication with the primary for network configuration of these containers
- A **Kubelet** is responsible for the running state of each node, ensuring that all containers on the node are healthy. It takes care of starting, stopping, and maintaining application containers organized into pods as directed by the control plane
- A **Kube-proxy** is an implementation of a network proxy and a load balancer, and it supports the service abstraction along with other networking operation. It is responsible for routing traffic to the appropriate container based on IP and port number of the incoming request

Kubernetes Pods

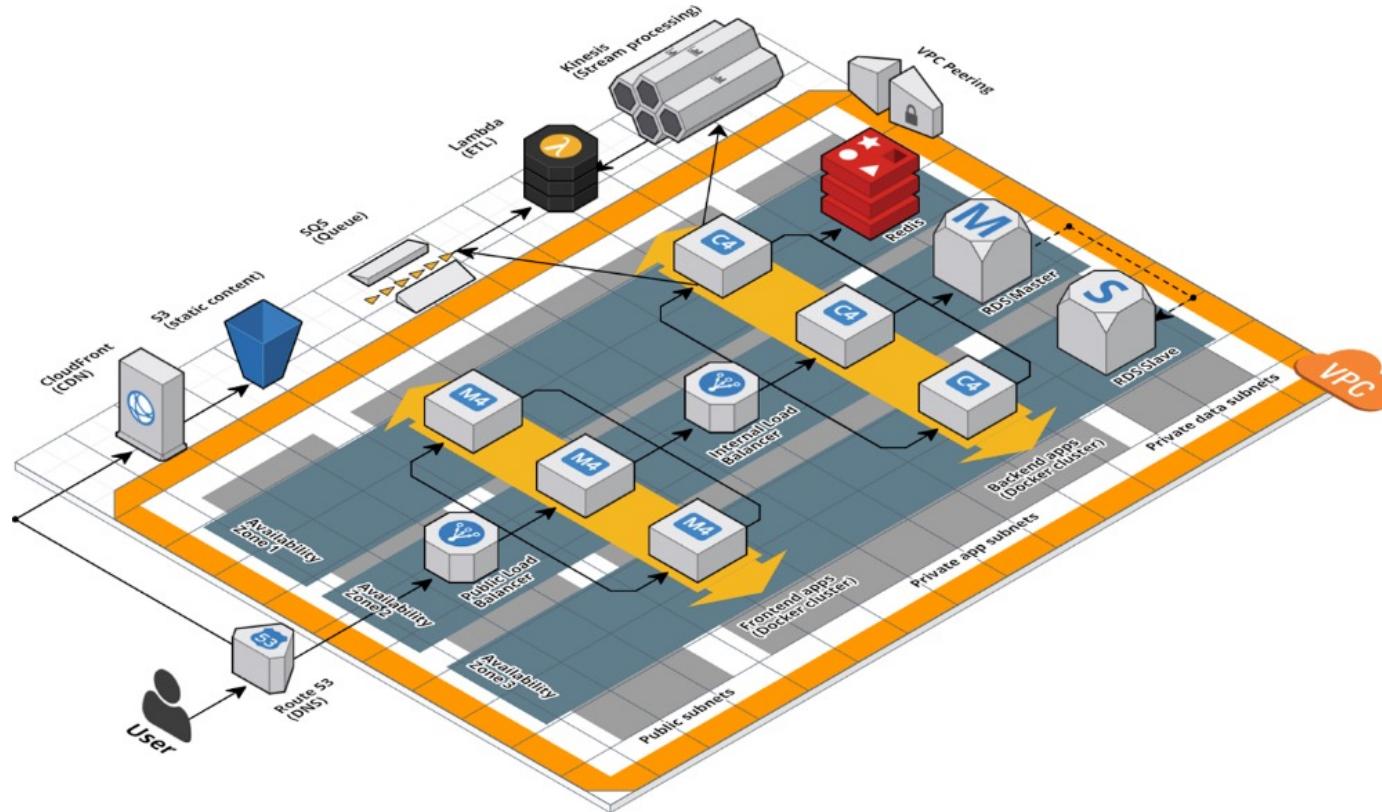
- A Pod represents a collection of application containers and volumes running in the same execution environment.
- A Pod is the smallest deployable artifact in a Kuberneeted cluster
- All of the containers in a Pod always land on the same machine
- Applications running in the same Pod share the same IP address and port space (network namespace), have the same hostname, and can communicate using native interprocess ccommunication channels
- On the opposite, applications in different Pods are isolated from each other; they have different IP addresses, different hostnames,...
- Containers in different Pods running on the same node might as well be on different servers
- In general, “**Will these containers work correctly if they land on different machines?**”
 - **No** -> use a Pod to group the containers
 - **Yes** -> use multiple Pods

Other option?

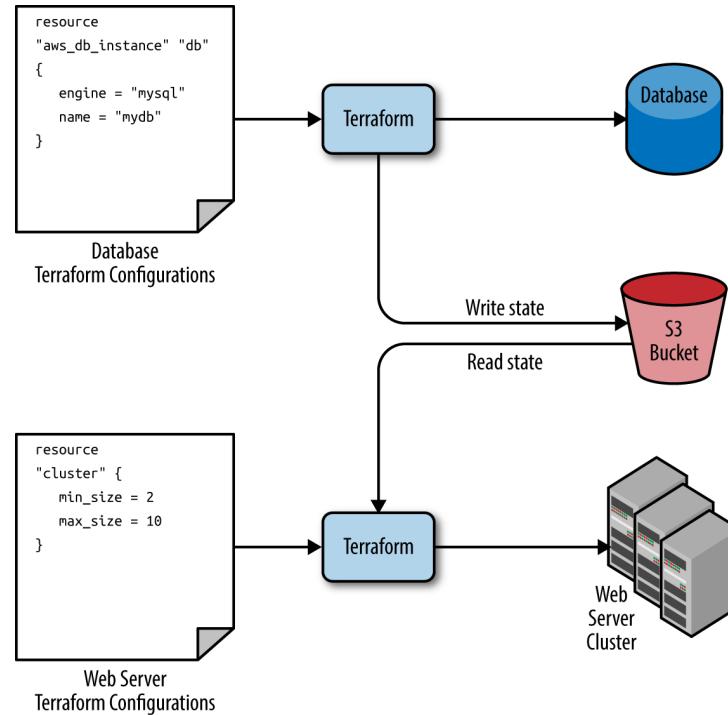
Terraform – Infrastructure as Code (IaC)

- Terraform is a solution to create cloud environments using code instead of using the cloud providers User Interfaces.
- Therefore, the management of the cloud resources takes less effort and the creation, or update, process is faster

Terraform – Infrastructure as Code (IaC)



Terraform – State sharing



But, I do not want to deploy the cluster, is it possible?

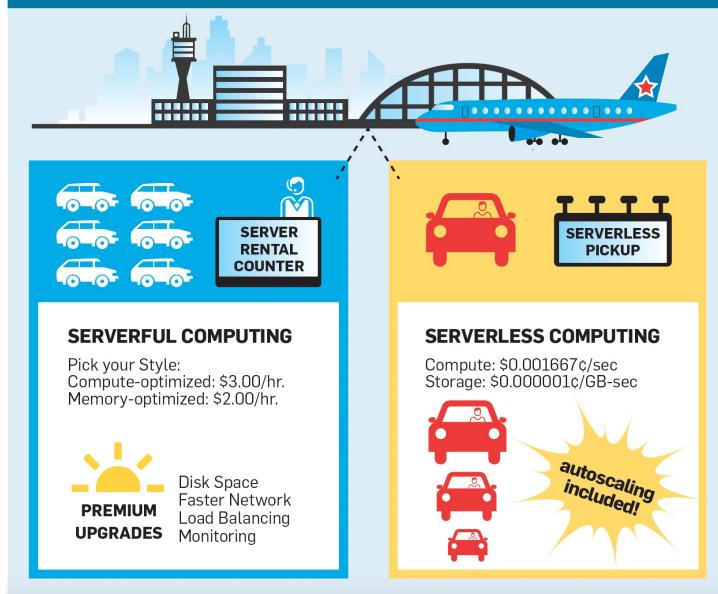
Serverless computing

- Serverless computing is a platform that hides server usage from developers and runs code on-demand automatically scaled and billed only for the time the code is running
- Cloud Native Computing Foundation (CNCF) defines serverless computing as *“the concept of building and running applications that do not require server management. It describes a finer grained deployment model where applications, bundled as one or more functions, are uploaded to a platform and then executed, scaled, and billed in response to the exact demand needed at the moment.”*

Serverless essential qualities

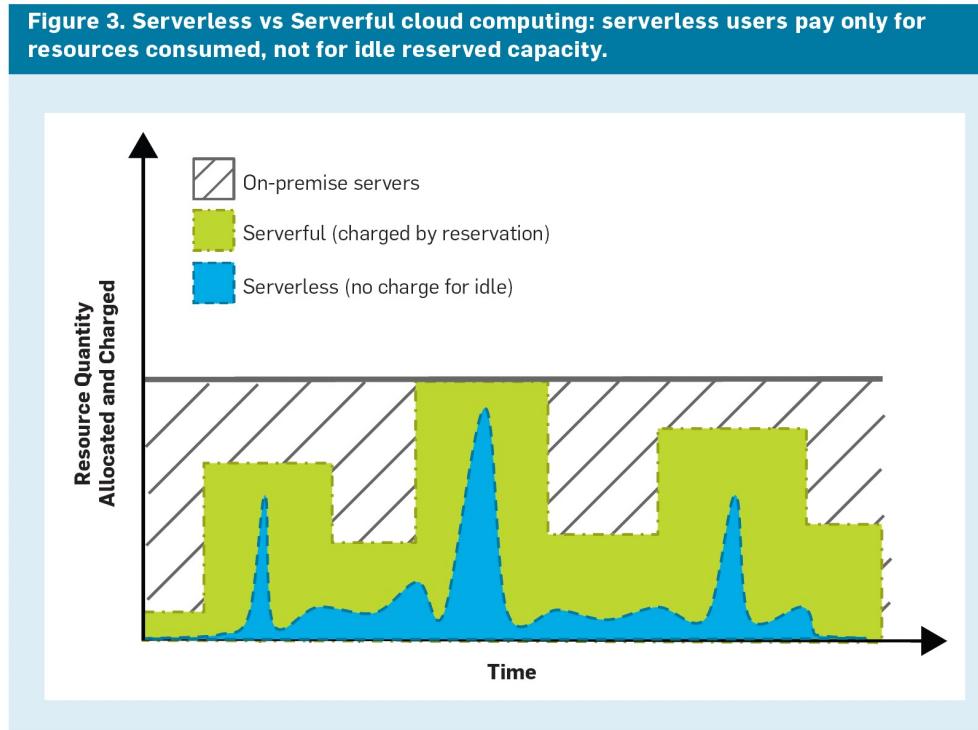
1. Providing an **abstraction** that hides the servers and the complexity of programming and operating them
2. Offering a **pay-as-you-go cost model** instead of a reservation-based model, so there is no charge for idle resources
3. **Elasticity - Automatic, rapid, and unlimited scaling** resources up and down to match demand closely, from zero to practically infinite

Figure 1. Cloud computing approaches compared to rides from an airport: Serverful as renting a car and serverless as taking a taxi ride.



Resource usage: Serverless compared with Serverful

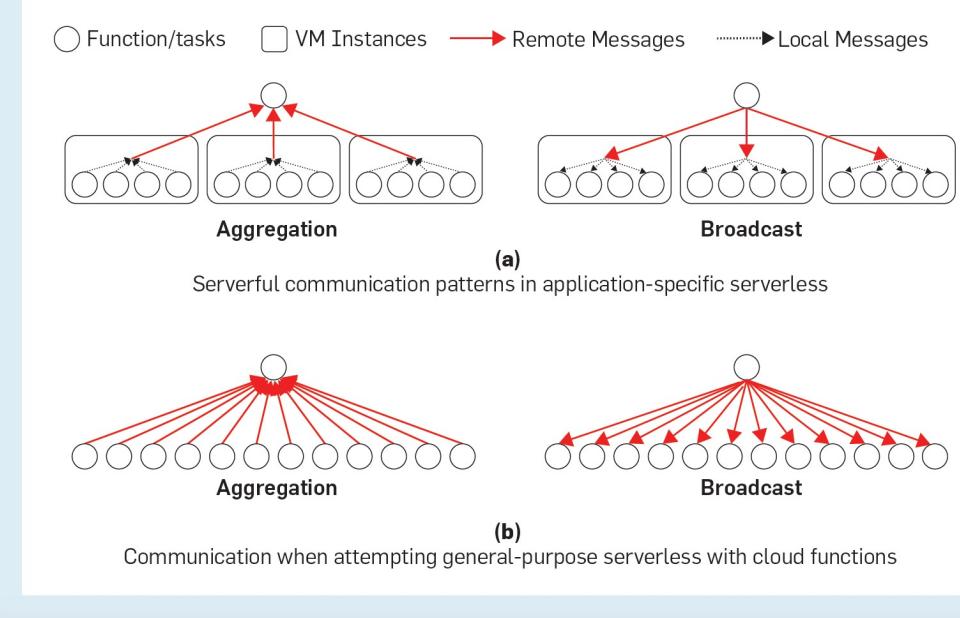
Figure 3. Serverless vs Serverful cloud computing: serverless users pay only for resources consumed, not for idle reserved capacity.



Increased communication with Serverless

Figure 4. Increased communication for aggregation and broadcast patterns.

Application-specific serverless frameworks (for example, Cloud Dataflow) can be implemented with serverful communication patterns. In this case (a) the fewer arrows indicate less network communication than in (b) the general-purpose serverless option. By packing K tasks per VM instance, an application-specific serverless solution, like a serverful solution, is able to achieve a communication complexity of $O(N/K)$ for a job with N tasks, as opposed to $O(N)$ for the cloud function based alternative which can not influence task placement. Typical values for K range from 10 to 100, leading to an overall difference of one to two orders of magnitude.



Function as a Service (FaaS)

- Function-as-a-Service is a serverless computing platform where the **unit of computation** is a function that is executed in response to triggers such as events or HTTP requests

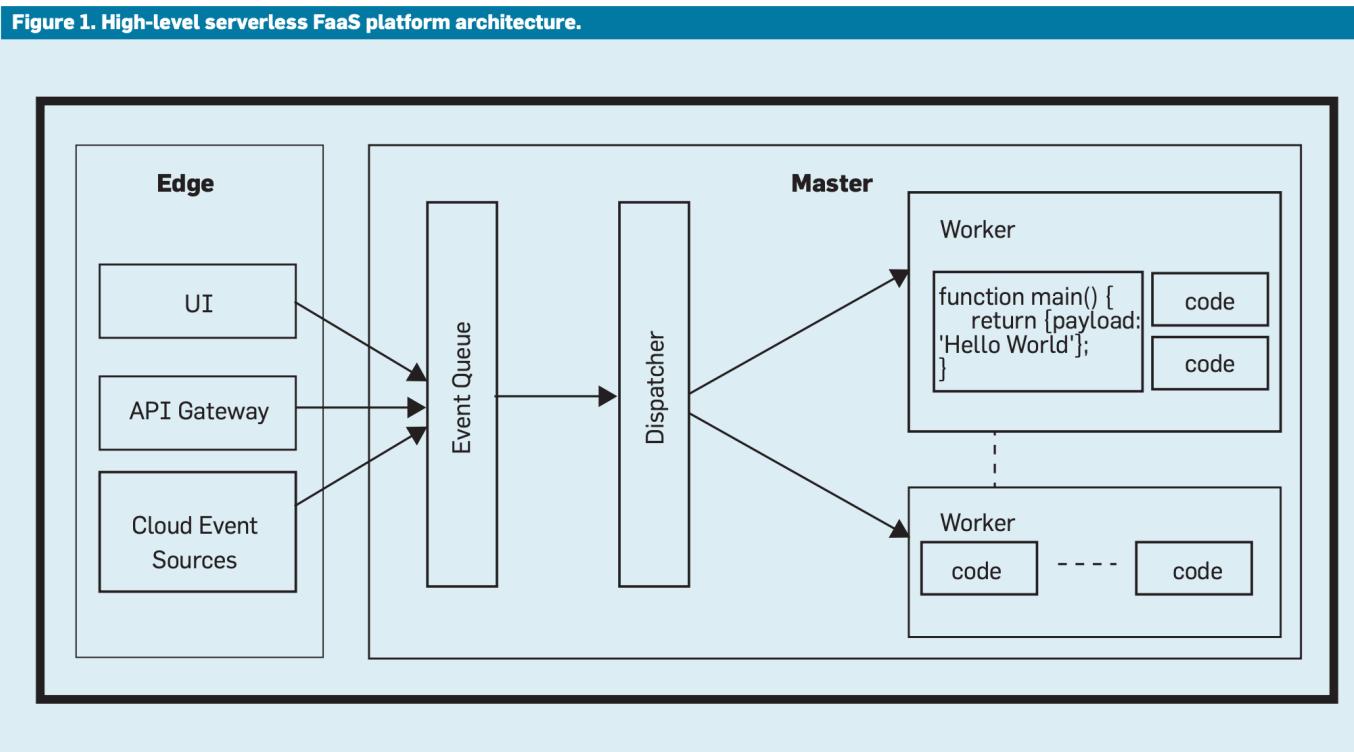
Table 1. Comparison of different choices for cloud as a service.

	IaaS	1st Gen PaaS	FaaS	BaaS/SaaS
Expertise required	High	Medium	Low	Low
Developer Control/Customization allowed	High	Medium	Low	Very low
Scaling/Cost	Requires high-level of expertise to build auto-scaling rules and tune them	Requires high-level of expertise to build auto-scaling rules and tune them	Auto-scaling to work load requested (function calls), and only paying for when running (scale to zero)	Hidden from users, limits set based on pricing and QoS
Unit of work deployed	Low-level infrastructure building blocks (VMs, network, storage)	Packaged code that is deployed and running as a service	One function execution	App-specific extensions
Granularity of billing	Medium to large granularity: minutes to hours per resource to years for discount pricing	Medium to large granularity: minutes to hours per resource to years for discount pricing	Very low granularity: hundreds of milliseconds of function execution time	Large: typically, subscription available based on maximum number of users and billed in months

BaaS – back-end as a service, e.g., object storage or databases

FaaS platform architecture

Figure 1. High-level serverless FaaS platform architecture.

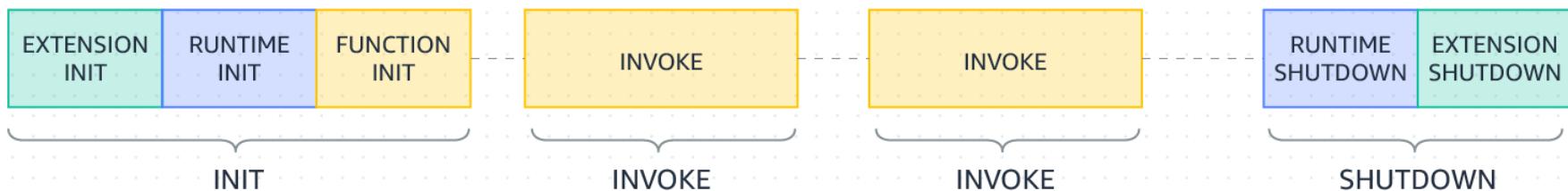


AWS Lambda functions

Function: is a resource that you can invoke to run your code in AWS Lambda. A function has code that processes events, and a runtime that passes requests and responses between Lambda and the function code. You provide the code, and you can use the provided runtimes or create your own.

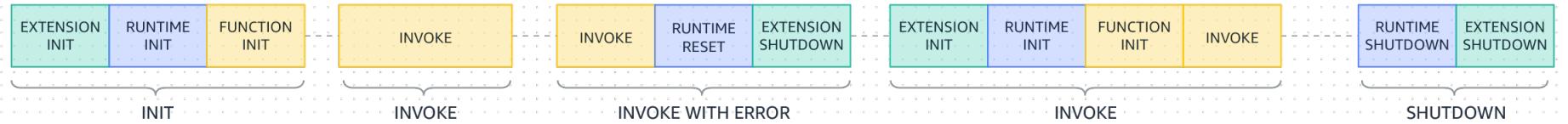
- **Runtime:** Lambda runtimes allow functions in different languages to run in the same base execution environment. You configure your function to use a runtime that matches your programming language. The runtime sits in between the Lambda service and your function code, relaying invocation events, context information, and responses between the two. You can use runtimes provided by Lambda or build your own.
- **Event:** is a JSON formatted document that contains data for a function to process. The Lambda runtime converts the event to an object and passes it to your function code. When you invoke a function, you determine the structure and contents of the event. When an AWS service invokes your function, the service defines the event.
- **Concurrency:** is the number of requests that your function is serving at any given time. When your function is invoked, Lambda provisions an instance of it to process the event. When the function code finishes running, it can handle another request. If the function is invoked again while a request is still being processed, another instance is provisioned, increasing the function's concurrency.
- **Trigger:** is a resource or configuration that invokes a Lambda function. This includes AWS services that can be configured to invoke a function, applications that you develop, and event source mappings. An event source mapping is a resource in Lambda that reads items from a stream or queue and invokes a function.

AWS Lambda functions, how does it works?



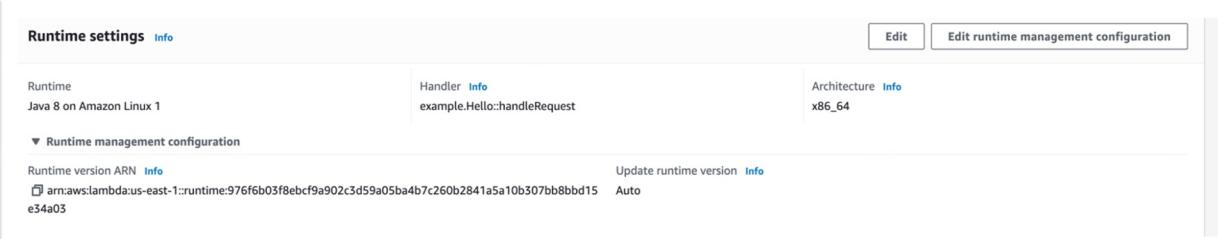
Cold start

Warm execution



AWS Lambda functions

Upload the code



Runtime settings [Info](#)

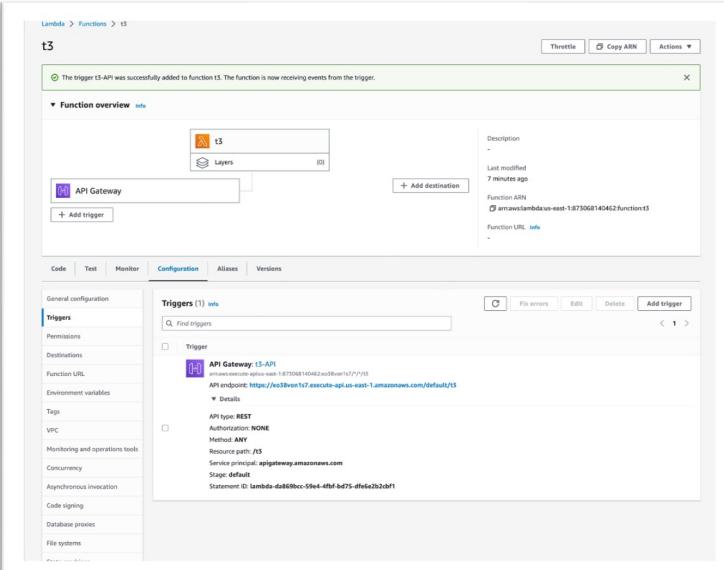
Runtime: Java 8 on Amazon Linux 1
Handler: [Info](#) example.Hello::handleRequest
Architecture: [Info](#) x86_64

Handler ARN: arn:aws:lambda:us-east-1:runtime:976f6b05f8ebcf9a902c3d59a05ba4b7c260b2841a5a10b307bb8bb15 e34a03

Runtime version ARN: [Info](#) arn:aws:lambda:us-east-1::runtime:976f6b05f8ebcf9a902c3d59a05ba4b7c260b2841a5a10b307bb8bb15 Auto

Update runtime version: [Info](#)

Define a trigger



Lambda > Functions > t3

The trigger t3-API was successfully added to function t3. The function is now receiving events from the trigger.

Function overview [Info](#)

t3

Description: Last modified 7 minutes ago
Function ARN: arn:aws:lambda:us-east-1:873068140462:function:t3
Function URL: [Info](#)

Code **Test** **Monitor** **Configuration** **Aliases** **Versions**

Triggers (1) [Info](#)

Trigger

API Gateway: t3-API
arn:aws:lambda:us-east-1:873068140462:trigger:t3/1/
API endpoint: <https://es3bevent17.execute-api.us-east-1.amazonaws.com/default/t3>

General configuration

- Triggers
- Permissions
- Destinations
- Function URL
- Environment variables
- Tags
- VPC
- Monitoring and operations tools
- Concurrency
- Asynchronous invocation
- Code signing
- Database proxies
- File systems

AWS Lambda functions

Test the function

APIs > /3 - ANY - Method Test

Method: PUT

Path: /3

No path parameters exist for this resource. You can define path parameters by using the syntax {pathParam} in a resource path.

Query Strings: #

Headers: #

Logs:

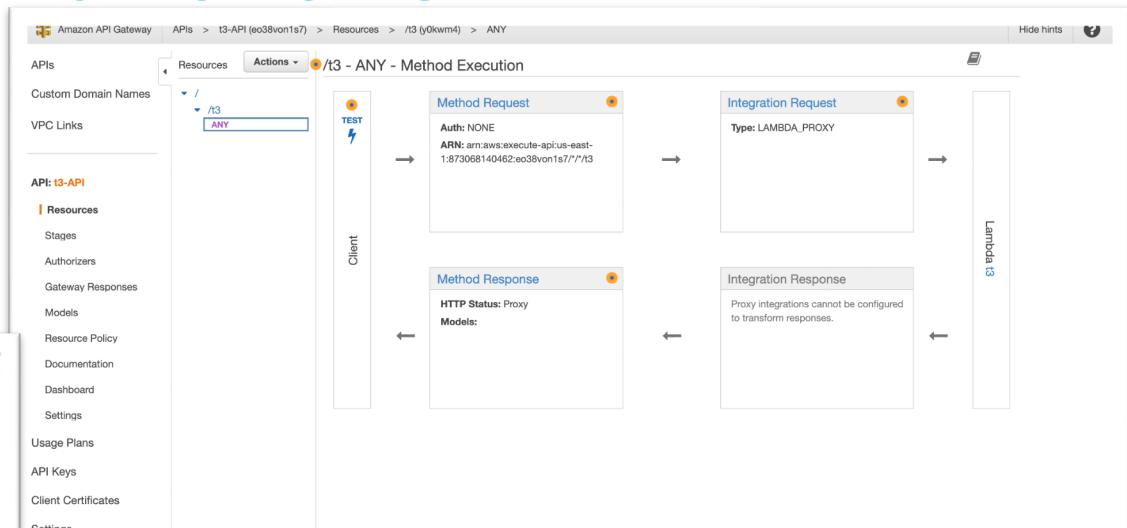
```
Execution log for request: ed4dded-9d2-411d-98d0-d980c5cc5dbb
Tue Feb 14 11:17:14 UTC 2023 - Starting execution for request: ed4dded-9d2-411d-98d0-d980c5cc5dbb
Tue Feb 14 11:17:14 UTC 2023 - Received request headers: {x-amz-date=20230214T111714Z, x-amz-sns-topic=arn:aws:sns:eu-central-1:12345678901234567890:lambda-test, x-amz-lambda-integration-request-id=ed4dded-9d2-411d-98d0-d980c5cc5dbb, x-amz-trace-id=d95e11-4ed9d4a-4bedf7330e440013ebf7123, x-amz-lambda-integration-taged=true}
Tue Feb 14 11:17:14 UTC 2023 - Received request body before transformations: {}
Tue Feb 14 11:17:14 UTC 2023 - Received request body after transformations: {"message": "Hello Integrator Impresarial!"}
```

Stage Variables: No stage variables exist for this method.

Client Certificates: No client certificates have been generated.

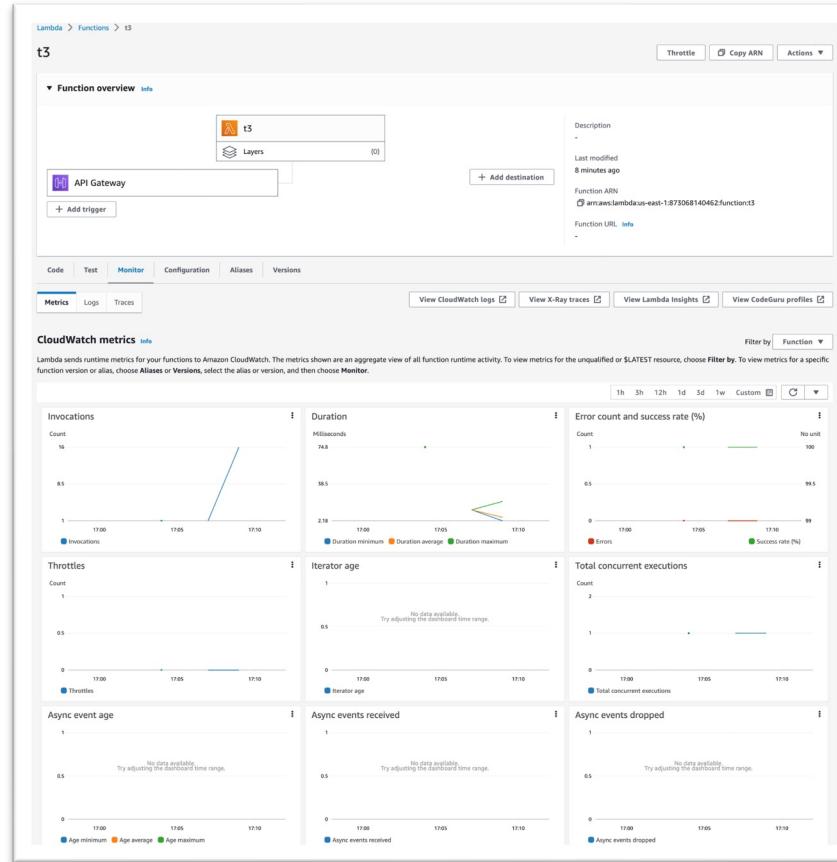
Request Body:

```
{ "name": "Integrator Impresarial" }
```



AWS Lambda functions

Check
consumption



Serverless drawbacks

- Not suitable in terms of cost and execution platform for long-running processes
- Vendor lock-in
- Cold starts can cause an overhead that is unacceptable on low-latency applications
- Monitoring and debugging is more complex when compared with other architectures

Software Bill of Materials (SBOM)

Bill of Materials (BOM)



LOW FAT VANILLA FLAVOURED YOGHURT

INGREDIENTS: Skim Milk, Concentrated Skim Milk, Water, Sugar, Cream (From Milk), Thickeners (1422 (From Maize), 1442 (From Maize)), Milk Solids, Gelatine, Flavours, Acidity Regulators (331, 332, 270, 330), Enzyme (Lactase), Live Cultures.

Contains Milk and Milk Products.



**The Minimum Elements
For a Software Bill of Materials (SBOM)**

Pursuant to
Executive Order 14028
on Improving the Nation's Cybersecurity

The United States Department of Commerce

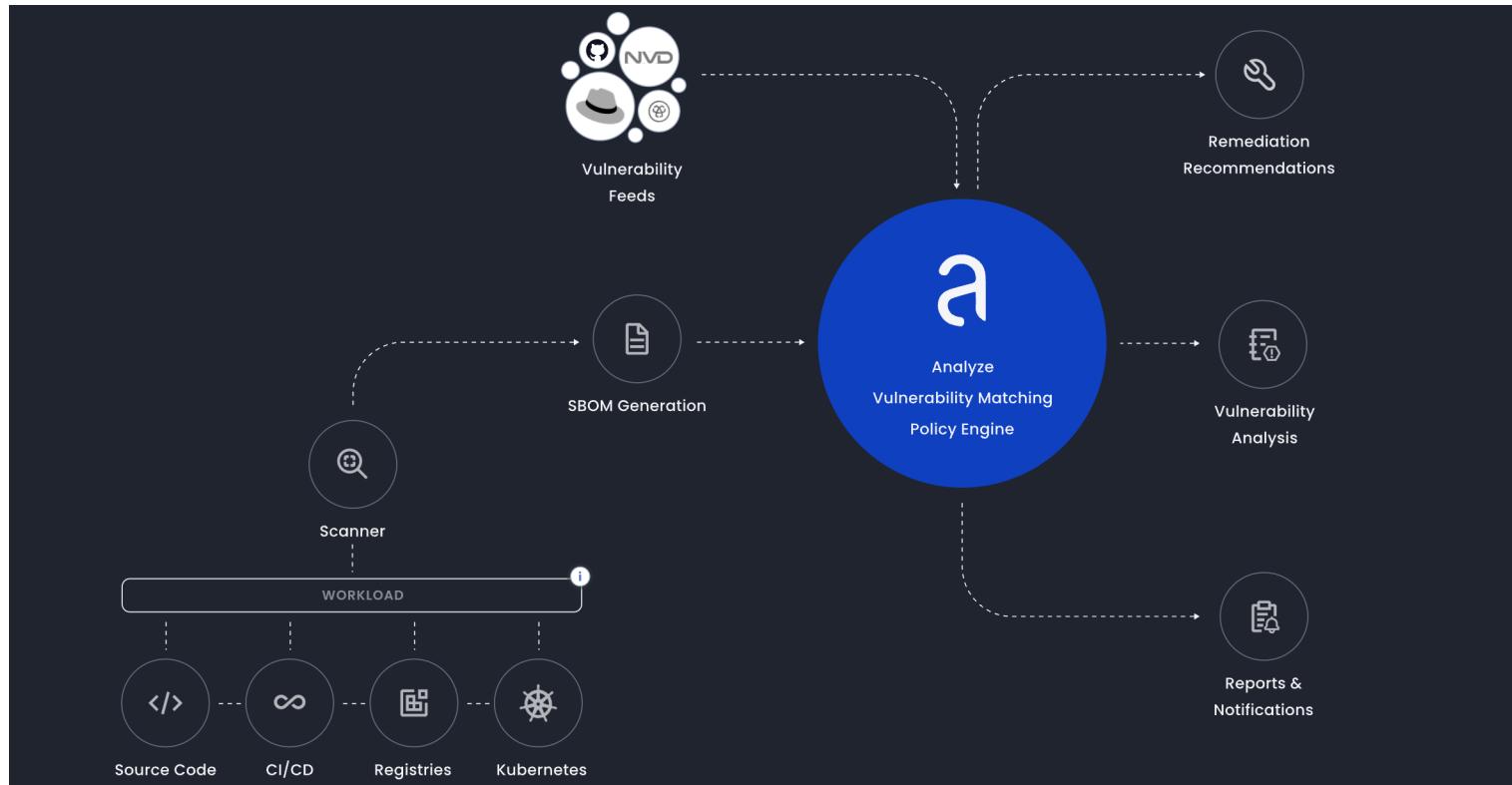
July 12, 2021

SBOM

https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

How Container Vulnerability Scanning works

<https://anchore.com/container-vulnerability-scanning/>



How Kubernetes Image Scanning works

<https://anchore.com/kubernetes/>



Preview of open-source available tools

<https://anchore.com/opensource/>

Syft



A CLI tool for generating a Software Bill of Materials (SBOM) from container images and filesystems.

[Try Syft](#)[Watch in action](#)

Grype



An easy-to-integrate open source vulnerability scanning tool for container images and filesystems.

[Try Grype](#)[Watch in action](#)

```
[sergioguerreiro@kdbio45 DockerFileKafka % docker build . -t kafka
[+] Building 40.8s (26/26) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 923B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [internal] load build context
=> => transferring context: 101.34MB
=> [ 1/21] FROM docker.io/library/ubuntu@sha256:20fa2d7bb4de7723f542be5923b06c4d704370f0390e4ae9e1c83
=> => resolve docker.io/library/ubuntu@sha256:20fa2d7bb4de7723f542be5923b06c4d704370f0390e4ae9e1c833c 0.0s
=> sha256:20fa2d7bb4de7723f542be5923b06c4d704370f0390e4ae9e1c833c8785644c1 1.42kB / 1.42kB 0.0s
=> => sha256:1bc0bc3815bdcfafefab3ef1d8fd159564693d0f8fb37b8151074651a11ffb 529B / 529B 0.0s
=> => sha256:21735dab04ba0ea14c23c491690a46073e0e6c72fdcc100f61bc610124eaa8c9 1.48kB / 1.48kB 0.0s
=> => sha256:00f50047d6061c27e70588a5aab89adada756e87d782a6c6bd08b4139eb8ea10 28.38MB / 28.38MB 0.7s
=> => extracting sha256:00f50047d6061c27e70588a5aab89adada756e87d782a6c6bd08b4139eb8ea10 1.4s
=> [ 2/21] RUN apt-get update
=> [ 3/21] RUN apt-get install -y sudo
=> [ 4/21] RUN sudo apt-get install -y openjdk-8-jdk
=> [ 5/21] RUN apt-get install -y telnet
=> [ 6/21] RUN apt-get install -y vim
=> [ 7/21] RUN apt-get install -y git
=> [ 8/21] RUN apt-get install -y maven
=> [ 9/21] COPY apache-zookeeper-3.8.0-bin.tar.gz /root
=> [10/21] COPY kafka_2.13-3.1.0.tgz /root
=> [11/21] COPY zoo.cfg /root
=> [12/21] COPY start.sh /root
=> [13/21] RUN tar -zxf /root/apache-zookeeper-3.8.0-bin.tar.gz
=> [14/21] RUN mv apache-zookeeper-3.8.0-bin /usr/local/zookeeper
=> [15/21] RUN mkdir -p /var/lib/zookeeper
=> [16/21] RUN mv /root/zoo.cfg /usr/local/zookeeper/conf
=> [17/21] RUN tar -zxf /root/kafka_2.13-3.1.0.tgz
=> [18/21] RUN mv kafka_2.13-3.1.0 /usr/local/kafka
=> [19/21] RUN mkdir /tmp/kafka-logs
=> [20/21] RUN echo "/usr/local/zookeeper/bin/zkServer.sh start" >> /root/.profile
=> [21/21] RUN echo "sudo /usr/local/kafka/bin/kafka-server-start.sh -daemon /usr/local/kafka/config/
=> exporting to image
=> => exporting layers
=> => writing image sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6 0.0s
=> => naming to docker.io/library/kafka 0.0s
```

% ./**syft** packages sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6 -o table >> output

- ✓ Loaded image
- ✓ Parsed image
- ✓ Cataloged packages [637 packages]

Human readable format

NAME	VERSION	TYPE
PyGObject	3.42.1	python
US_export_policy		java-archive
activation	1.1.1	java-archive
adduser	3.118ubuntu5	deb
adwaita-icon-theme	41.0-1ubuntu1	deb
alsa-topology-conf	1.2.5.1-2	deb
alsa-ucm-conf	1.2.6.3-1ubuntu1	deb
aopalliance	1.0	java-archive
aopalliance-repackaged	2.6.1	java-archive
apt	2.4.7	deb
argparse4j	0.7.0	java-archive
at-sp1-core	2.44.0-3	deb
atinject-jsr330-api	1.0	java-archive
audience-annotations	0.12.0	java-archive
audience-annotations	0.5.0	java-archive
base-files	12ubuntu4.2	deb
base-passwd	3.5.52build1	deb
bash	5.1-6ubuntu1	deb
bsdutils	1:2.37.2-4ubuntu3	deb
ca-certificates	20211016	deb
ca-certificates-java		java-archive
ca-certificates-java	20190909	deb
cdi-api	1.2	java-archive
cgleib	3.2.12	java-archive
charsets		java-archive
cldrdata		java-archive
commons-cli	1.4	java-archive
commons-codec	1.14	java-archive
commons-io	2.11.0	java-archive
commons-io	2.6	java-archive
commons-lang3	3.11	java-archive
commons-lang3	3.8.1	java-archive
connect-api	3.1.0	java-archive
connect-basic-auth-extension	3.1.0	java-archive
connect-file	3.1.0	java-archive
connect-json	3.1.0	java-archive
connect-mirror	3.1.0	java-archive
connect-mirror-client	3.1.0	java-archive
connect-runtime	3.1.0	java-archive
connect-transforms	3.1.0	java-archive
coreutils	8.32-4.1ubuntu1	deb
dash	0.5.11+git20210903+057cd650a4ed-3build1	deb
dbus	1.12.20-2ubuntu4	deb
dbus-python	1.2.18	python
dbus-user-session	1.12.20-2ubuntu4	deb
dconf-gsettings-backend	0.40.0-3	deb
dconf-service	0.40.0-3	deb
debconf	1.5.79ubuntu1	deb
debianutils	5.5-1ubuntu2	deb
diffutils	1:3.8-0ubuntu2	deb
dmsetup	2:1.02.175-2.1ubuntu4	deb
dnsns		java-archive
dpkg	1.21.1ubuntu2.1	deb
dt		java-archive
e2fsprogs	1.46.5-2ubuntu1.1	deb

syft — more output — 163x60

```
{
    "version": 0,
    "job": {},
    "detector": {
        "name": "syft",
        "url": "https://github.com/anchore/syft",
        "version": "0.56.0"
    },
    "metadata": {
        "syft:distro": "pkg:generic/ubuntu@22.04?like=debian"
    },
    "manifests": {
        "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/charsets.jar": {
            "name": "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/charsets.jar",
            "file": {
                "source_location": "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/charsets.jar"
            },
            "metadata": {
                "syft:filesystem": "sha256:cd02f67aba6a5a352784d578a08e7cfbee45571907658fd1ce9b5d29d36ea3f6"
            },
            "resolved": {
                "pkg:maven/charsets/charsets": {
                    "package_url": "pkg:maven/charsets/charsets",
                    "relationship": "direct",
                    "scope": "runtime"
                }
            }
        },
        "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/ext/cldrdata.jar": {
            "name": "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/ext/cldrdata.jar",
            "file": {
                "source_location": "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/ext/cldrdata.jar"
            },
            "metadata": {
                "syft:filesystem": "sha256:cd02f67aba6a5a352784d578a08e7cfbee45571907658fd1ce9b5d29d36ea3f6"
            },
            "resolved": {
                "pkg:maven/cldrdata/cldrdata": {
                    "package_url": "pkg:maven/cldrdata/cldrdata",
                    "relationship": "direct",
                    "scope": "runtime"
                }
            }
        },
        "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/ext/dnsns.jar": {
            "name": "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/ext/dnsns.jar",
            "file": {
                "source_location": "sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dfffc2f78e2fadfaaf7b6:/lib/jvm/java-1.8.0-openjdk-arm64/jre/lib/ext/dnsns.jar"
            },
            "metadata": {
                "syft:filesystem": "sha256:cd02f67aba6a5a352784d578a08e7cfbee45571907658fd1ce9b5d29d36ea3f6"
            },
            "resolved": {
                "pkg:maven/dnsns/dnsns": {
                    "package_url": "pkg:maven/dnsns/dnsns",
                    "relationship": "direct",
                    "scope": "runtime"
                }
            }
        }
    }
}
```

output

JSON format

% ./grype sha256:ce774c2bb577d868ec312839d63b9f4545ec20027a69dffc2f78e2fadfaaf7b6

- ✓ Vulnerability DB
- ✓ Loaded image
- ✓ Parsed image
- ✓ Cataloged packages
- ✓ Scanned image

[updated]

[637 packages]

[444 vulnerabilities]



NAME	INSTALLED	FIXED-IN	TYPE	VULNERABILITY	SEVERITY
commons-io	2.6		java-archive	CVE-2021-29425	Medium
commons-io	2.6	2.7	java-archive	GHSA-gwrrp-pvraq-jmwv	Medium
coreutils	8.32-4.1ubuntu1		deb	CVE-2016-2781	Low
geronimo-annotation_1.3_spec	1.3		java-archive	CVE-2008-0732	Low
geronimo-interceptor_3.0_spec	1.3		java-archive	CVE-2011-5034	High
geronimo-interceptor_3.0_spec	1.0.1		java-archive	CVE-2008-0732	Low
git	1:2.34.1-1ubuntu1.4		java-archive	CVE-2011-5034	High
git-man	1:2.34.1-1ubuntu1.4		deb	CVE-2018-1000021	Low
guava	29.0-jre		java-archive	CVE-2020-8998	Low
guava	29.0-jre		java-archive	GHSA-5mg8-w23w-74h3	Low
httpclient	4.5.11		java-archive	CVE-2020-13956	Medium
httpclient	4.5.11	4.5.13	java-archive	GHSA-7r82-z7xv7-xcpj	Medium
jackson-databind	2.12.3	2.12.6.1	java-archive	GHSA-57j2-w4cx-62h2	High
jackson-databind	2.13.1	2.13.2.1	java-archive	GHSA-57j2-w4cx-62h2	High
jackson-databind	2.12.3		java-archive	CVE-2020-36518	High
jackson-databind	2.13.1		java-archive	CVE-2020-36518	High
jetty-client	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-client	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-continuation	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-continuation	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-httP	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-htp	9.4.43.v20210629	9.4.47	java-archive	GHSA-cj7v-27pg-wf7q	Low
jetty-htp	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-io	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-io	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-security	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-security	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-server	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-server	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-servlet	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-servlet	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-servlets	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-servlets	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-util	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jetty-util	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-util-ajax	9.4.43.v20210629		java-archive	CVE-2022-2048	High
jetty-util-ajax	9.4.43.v20210629		java-archive	CVE-2022-2047	Low
jsoup	1.10.2	1.14.2	java-archive	GHSA-m72m-nhh2-9p6c	High
jsoup	1.10.2		java-archive	CVE-2022-36033	Medium
jsoup	1.10.2	1.15.3	java-archive	GHSA-gp7f-rwxc-9369	Medium
libapparmor1	3.0.4-2ubuntu2.1		deb	CVE-2021-37714	High
libc-bin	2.35-0ubuntu3.1		deb	CVE-2016-1585	Medium
libc6	2.35-0ubuntu3.1		deb	CVE-2016-20013	Negligible
libcairo-gobject2	1.16.0-5ubuntu2		deb	CVE-2016-20013	Negligible
libcairo-gobject2	1.16.0-5ubuntu2		deb	CVE-2019-6461	Low
libcairo2	1.16.0-5ubuntu2		deb	CVE-2018-18064	Low
libcairo2	1.16.0-5ubuntu2		deb	CVE-2017-7475	Low
libcairo2	1.16.0-5ubuntu2		deb	CVE-2019-6461	Low
libcairo2	1.16.0-5ubuntu2		deb	CVE-2018-18064	Low
libcairo2	1.16.0-5ubuntu2		deb	CVE-2017-7475	Low

cve.org/CVERecord?id=CVE-2021-26291

CVE About Partner Information Program Organization Downloads Resources & Support

CVE-2021-26291 Find

Find CVE Records by keyword on cve.mitre.org ↗

❶ Welcome to the new CVE Beta website! [CVE List keyword search ↗](#) & [downloads ↗](#) will be temporarily hosted on the old [cve.mitre.org ↗](#) website use the [CVE Program web forms ↗](#) for any comments or concerns.

CVE-2021-26291 Detail

The CVE Record information displayed on this page may not be displaying the full range of available information due to differences in how the data may have been entered. If you feel that the information being displayed is not meeting your expectations, please let us know by using this [feedback form ↗](#).

View full JSON 4.0 record



Description	Apache Maven will follow repositories that are defined in a dependency's Project Object Model (pom) which may be surprising to some users, resulting in potential risk if a malicious actor takes over that repository or is able to insert themselves into a position to pretend to be that repository. Maven is changing the default behavior in 3.8.1+ to no longer follow http (non-SSL) repository references by default. More details available in the referenced urls. If you are currently using a repository manager to govern the repositories used by your builds, you are unaffected by the risks present in the legacy behavior, and are unaffected by this vulnerability and change to default behavior. See this link for more information about repository management: https://maven.apache.org/repository-management.html
State	PUBLIC
Problem Types	<ul style="list-style-type: none">Unexpected Behavior

Q&A





TÉCNICO LISBOA

Enterprise Integration

Business Process Management

Prof. Sérgio Guerreiro

Sergio.guerreiro@tecnico.ulisboa.pt

Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

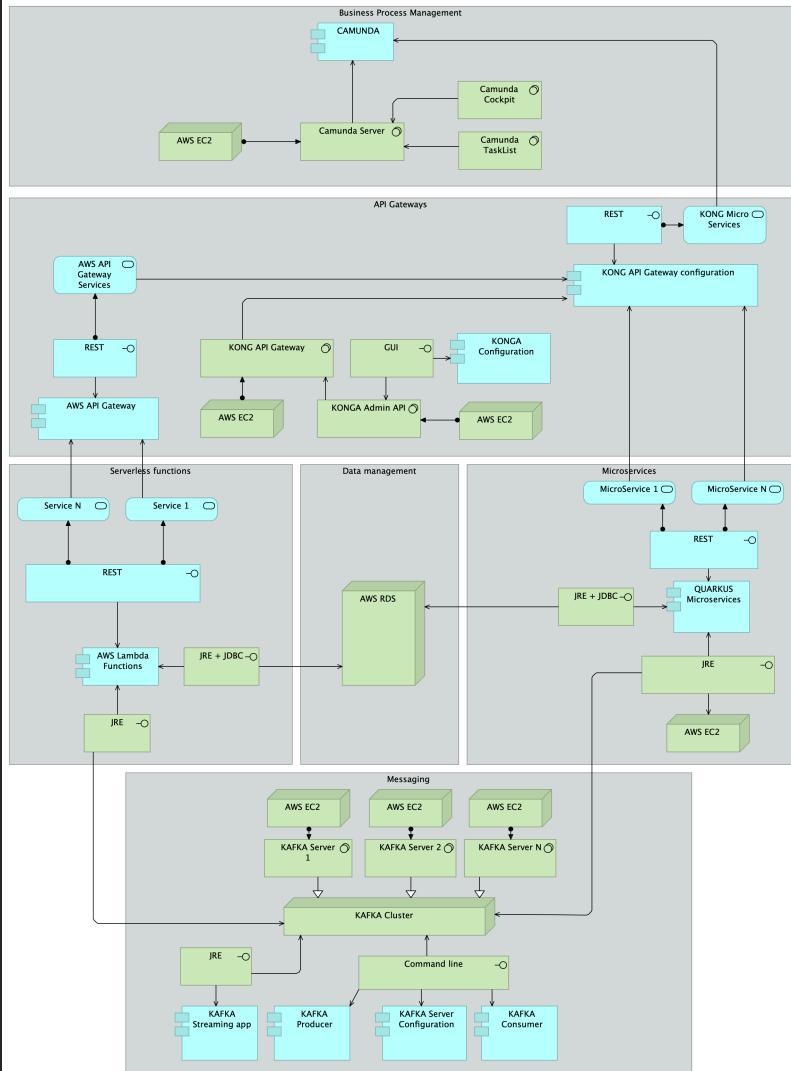
Camunda Engine is the business process executor that consumes the available microservices in the enterprise.

The goal is to enable the integration with all the remaining enterprise systems and to enable the representation and execution of the sequence of business behaviours in an easy way.

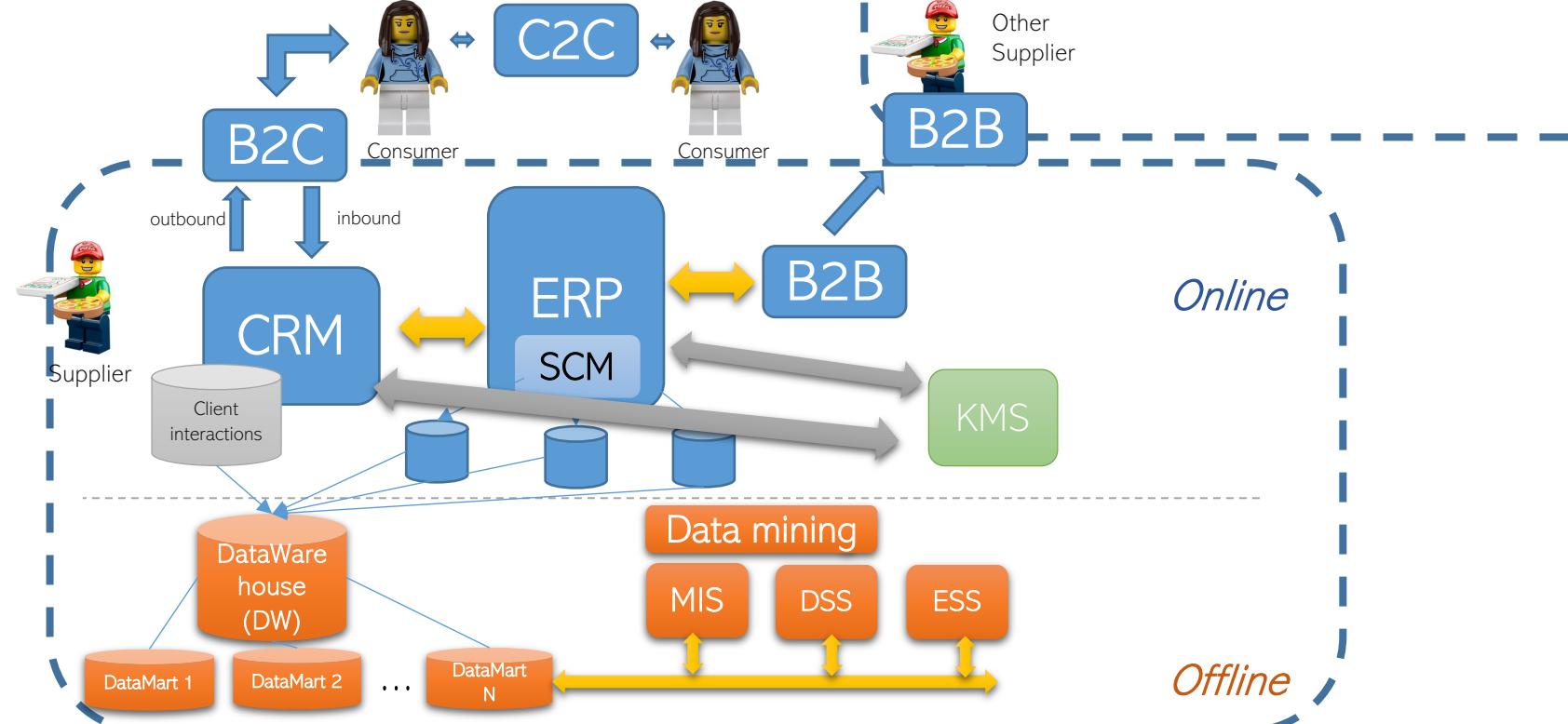
The process-oriented nature allows a closer approach to the business development of the enterprise.

IE Technology Stack

Overall view



ERP as the placeholder for Business Processes

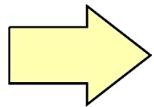


The precursors: the Workflow systems

- Workflow Management System is a **software platform** that supports the design, development, implementation and analysis of workflows.
- Create new applications as an exercise of composition (existing functions) **rather than the traditional development** of a new application by conventional programming
- Programming based on **models**, represented as straight forward visual constructs (visual programming)
- Workflows were considered, since the 90's also as a **strategic approach for application integration**
- For all of the above reasons considered as: **programming in the large**

Then, What is a business process?

Business process →



8.3.1 Business Process

A business process represents a sequence of business behaviors that achieves a specific result such as a defined set of products or business services.

A business process describes the internal behavior performed by a business role that is required to produce a set of products and services. For a consumer, the products and services are relevant and the required behavior is merely a black box, hence the designation “internal”.

A complex business process may be an aggregation of other, finer-grained processes. To each of these, finer-grained roles may be assigned.

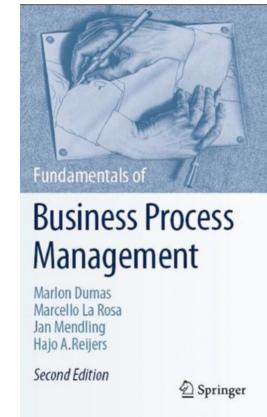
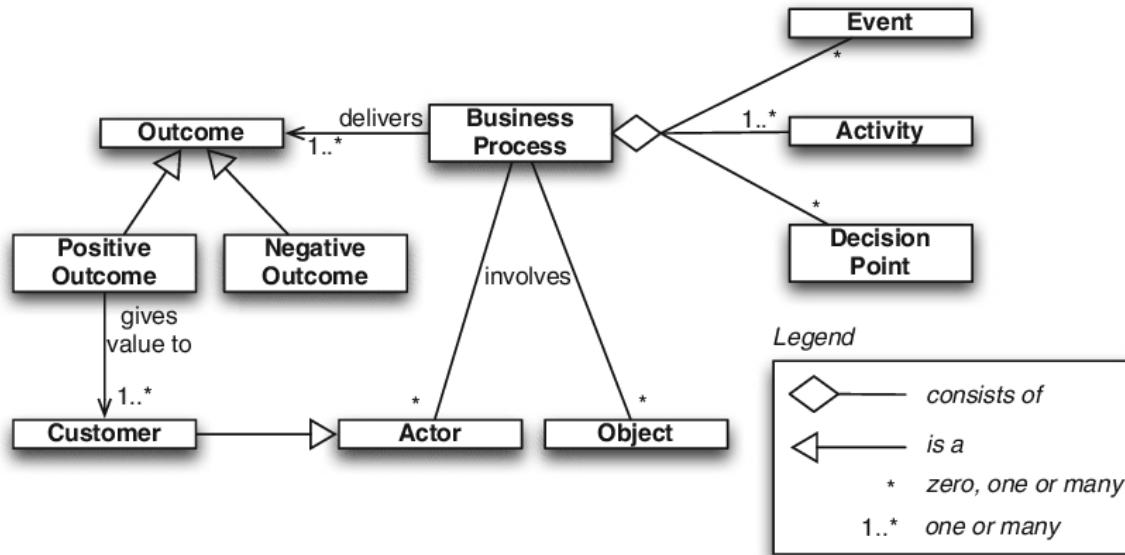
There is a potential many-to-many relationship between business processes and business functions. Informally speaking, processes describe some kind of “flow” of activities, whereas functions group activities according to required skills, knowledge, resources, etc.

A business process may be triggered by, or trigger, any other business behavior element (e.g., business event, business process, business function, or business interaction). A business process may access business objects. A business process may realize one or more business services and may use (internal) business services or application services. A business role may be assigned to a business process to perform this process manually. An automated business process can be realized by an application process. The name of a business process should clearly indicate a predefined sequence of actions using a verb or verb-noun combination and may include the word “process”. Examples are “adjudicate claim”, “employee on-boarding”, “approval process”, or “financial reporting”.

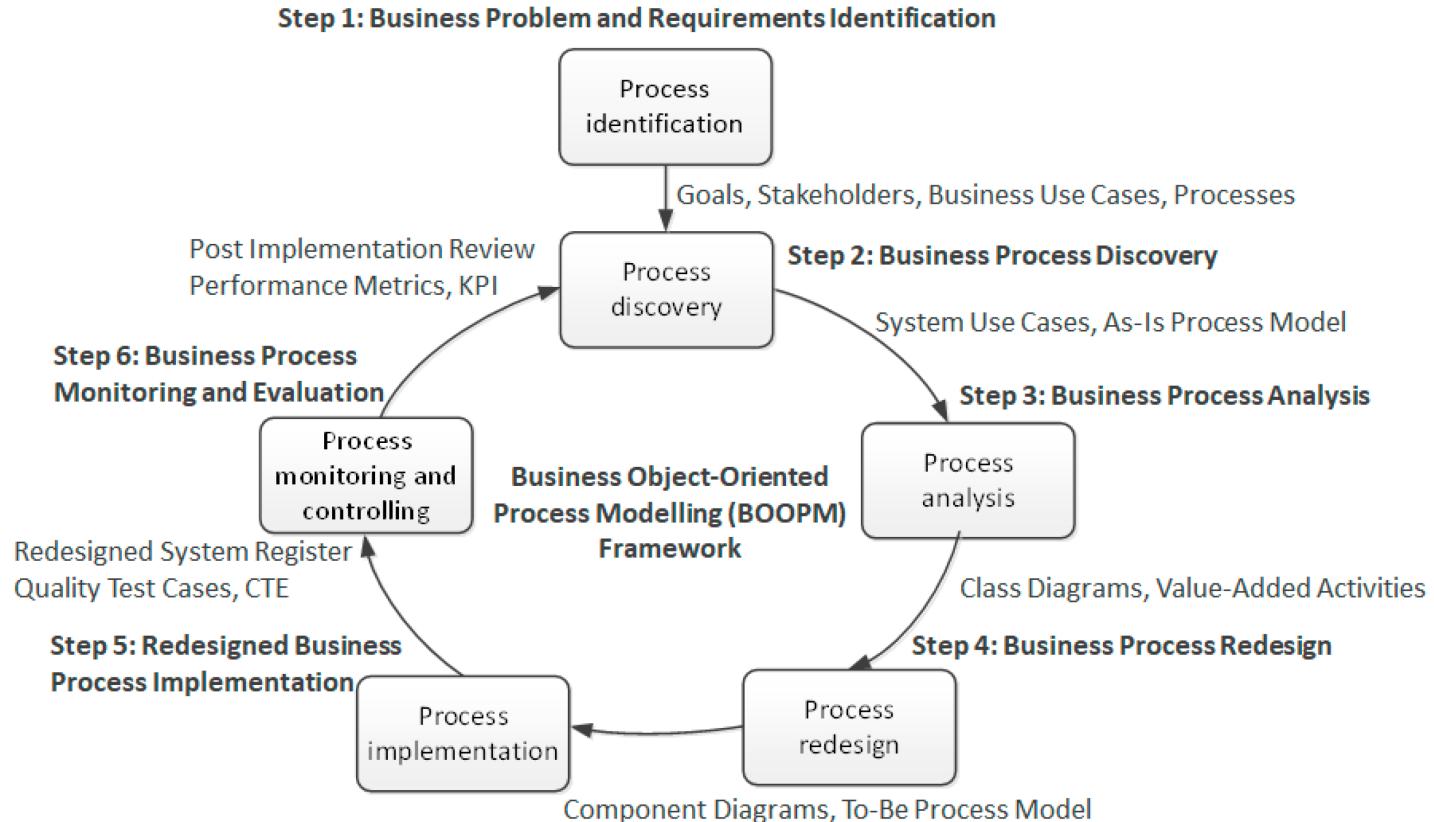
In an ArchiMate model, the existence of business processes is depicted. High-level business, end-to-end processes, macro flows, and workflows can all be expressed with the same business process element in the ArchiMate language. It does not, however, list the flow of activities in detail. This is typically done during business process modeling, where a business process can be expanded using a business process design language; e.g., BPMN [12].

Business Process definition

- Set of **interrelated activities** that transform **inputs** into **outputs** in order to produce a **service or product to a specific customer**.
- The focus is always on the final product!!!



A (classic) reference framework for the BPM lifecycle



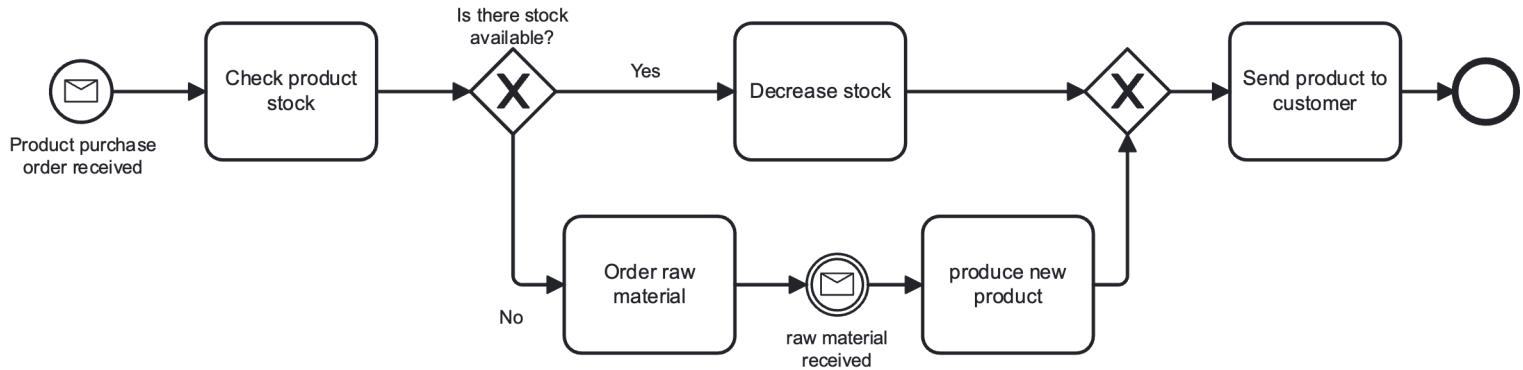
On “Process” versus “Business Process”

Process:

- “A set of interrelated and cooperative activities that transform inputs into outputs”. (ISO 9000)

Business Process:

- “A collection of activities that takes one or more kinds of input and creates an output **that is of value to the customer.**” (Hammer & Champy 1993)



Hello World! BPMN

When a seller receives a product purchase order, he always checks the available stock.

If there is stock available, update it and send the product to the customer.

If there is no stock available, order raw material and wait for it to be received. Then the new product is produced and sent to the customer.

“Business” Process Model and Notation (BPMN)

- BPMN is a OMG notation for Business Process Modelling.
- BPMN allows the graphical specification of business processes.
- BPMN can be mapped to execution languages and SOA environments...
- Timeline:
 - 2004: BPMI (Business Process Management Initiative) publishes BPMN 1.0
 - 2005: BPMI is integrated into OMG (Object Management Group)
 - 2006: OMG officially adopted the notation and publishes BPMN 1.0
- 2008: BPMN 1.2 published
- 2009: BPMN 2.0 draft release
- 2011: BPMN 2.0 published - www.omg.org/spec/BPMN/2.0/



<http://www.omg.org/bpmn/>

...all about BPMN: <http://www.bpmn.org/>

Modelling in BPMN – Process types

2.1 Process Modeling Conformance

The next eight sections describe **Process Modeling Conformance**.

2.1.1 BPMN Process Types

The implementations claiming **Process Modeling Conformance** MUST support the following **BPMN** packages:

- ◆ The **BPMN** core elements, which include those defined in the *Infrastructure*, *Foundation*, *Common*, and *Service* packages (see Chapter 8).
- ◆ **Process** diagrams, which include the elements defined in the **Process**, **Activities**, **Data**, and **Human Interaction** packages (see Chapter 10).
- ◆ **Collaboration** diagrams, which include **Pools** and **Message Flow** (see Chapter 9).
- ◆ **Conversation** diagrams, which include **Pools**, **Conversations**, and **Conversation Links** (see Chapter 9).

As an alternative to full **Process Modeling Conformance**, there are three conformance sub-classes defined:

- ◆ **Descriptive**
- ◆ **Analytic**
- ◆ **Common Executable**

Descriptive is concerned with visible elements and attributes used in high-level modeling. It should be comfortable for analysts who have used BPA flowcharting tools.

Analytic contains all of **Descriptive** and in total about half of the constructs in the full **Process Modeling Conformance** Class. It is based on experience gathered in BPMN training and an analysis of user-patterns in the Department of Defense Architecture Framework and planned standardization for that framework.

Both **Descriptive** and **Analytic** focus on visible elements and a minimal subset of supporting attributes/elements.

Common Executable focuses on what is needed for executable process models.

Elements and attributes not in these sub-classes are contained in the full **Process Modeling Conformance** class.

The elements for each sub-class are defined in the next section.

Types of BPMN Diagrams

- **Process**

- Represents the public or private **processes** of a participant.
- Focus on representing the (internal) **orchestration** of a process.
- The participant can be subdivided into multiple Lanes.
- All external pools (if any) must be black-box.

- **Collaboration**

- Represents the **message** exchange between **two or more participants**.
- Focus on representing the **orchestration** of a **process** across multiple **participants**.

- **Choreography**

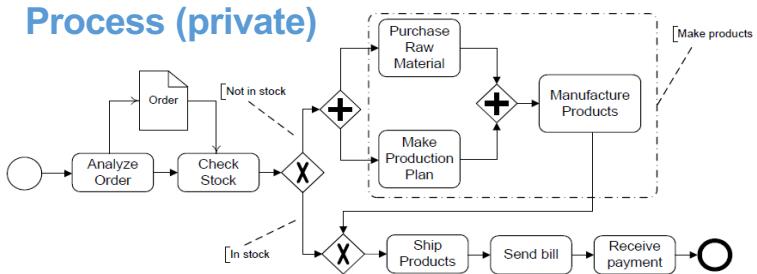
- Represent the information pertaining to each **participant** in the choreography.
- The focus is not on orchestrations of the work performed within these Participants, but rather on the exchange of information between Participants.

- **Conversation**

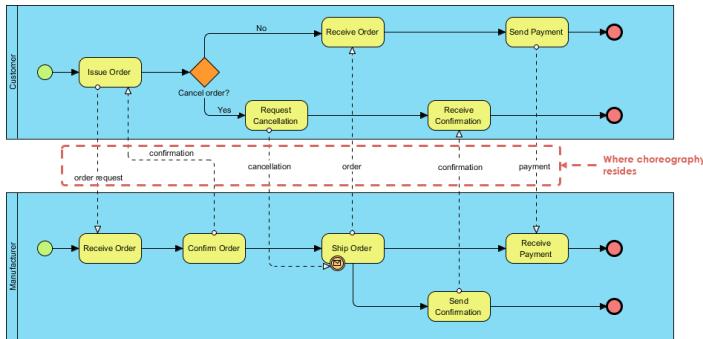
- Conversation diagrams visualize messages exchange between pools.
- Design workflow with business process diagram and visualize communications with BPMN conversation diagrams.

Types of BPMN Diagrams

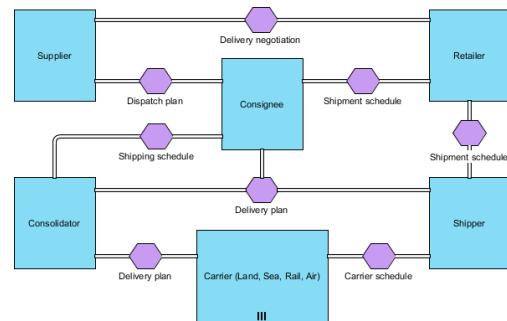
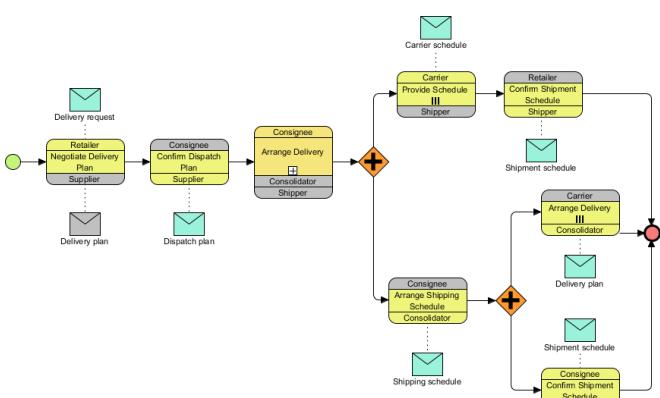
Process (private)



Collaboration



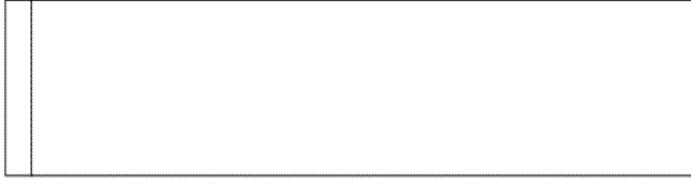
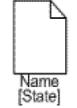
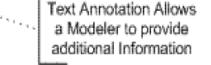
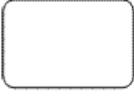
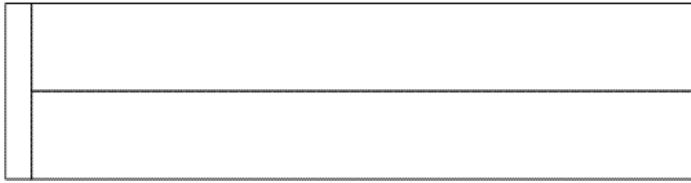
Choreography



Conversation

Core BPMN Elements

Core Set of BPMN Elements

Flow Objects	Connecting Object	Swimlanes	Artifacts
Events 	Sequence Flow 	Pool 	Data Object  Text Annotation 
Activities 	Message Flow 	Lanes (within a Pool) 	
Gateways 	Association 		Group 

Pools and Lanes

Pool: represents a process Participant.

Sequence flows **cannot cross Pool** boundaries

Message flows can cross **Pool** boundaries

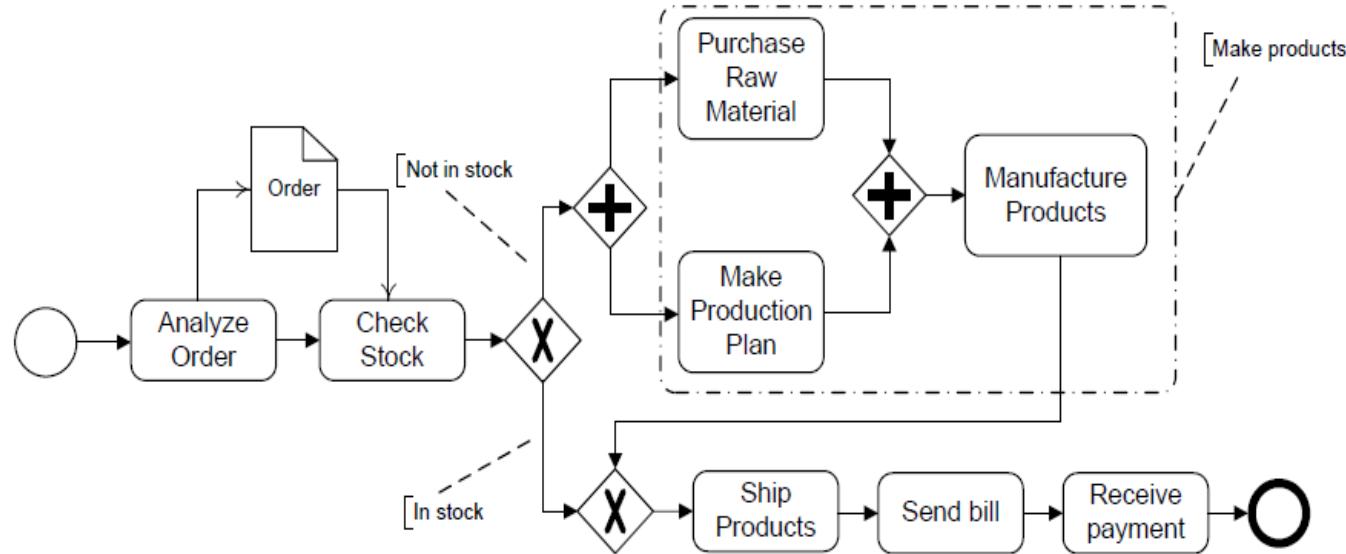


Lane: a sub-division of a Pool. Used to organize and categorize the activities of a Participant.

Sequence flows can cross **Lane** boundaries



Private Process Diagram (without pools)



- A **Process** focuses on a single **Participant**.
- A **Private Process** focuses on a **Participant** internal to the organization.

Private Process with one Pool and nested Lanes

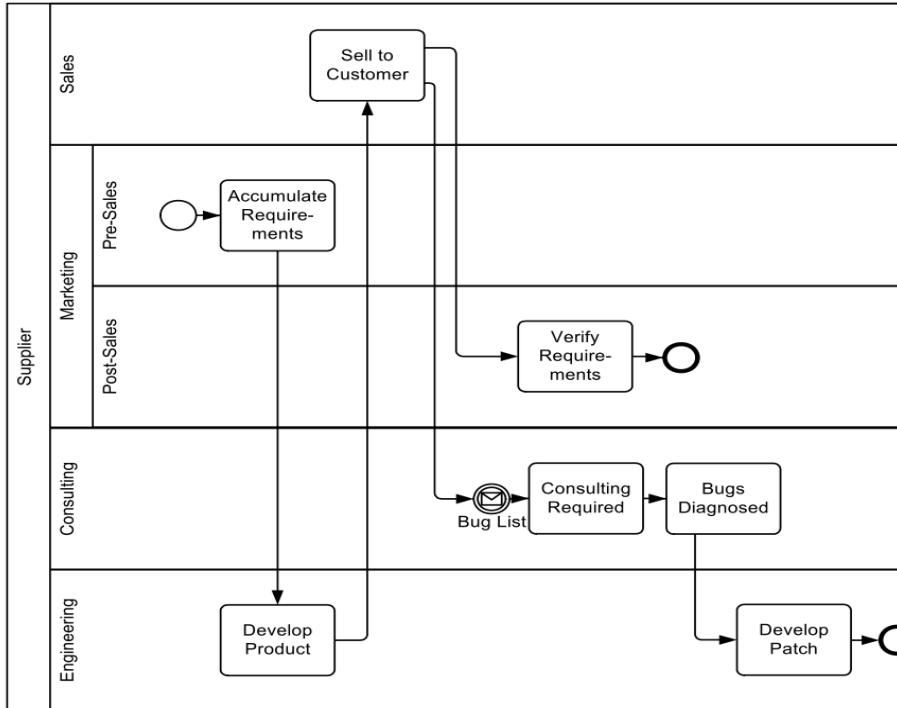


Figure 10.125 - An Example of Nested Lanes

This **Process** focuses on a single **Participant** with multiple **Lanes**. Therefore it is a **private Process**.

Public vs. Private Process



Figure 7.1 - Example of a *private* Business Process

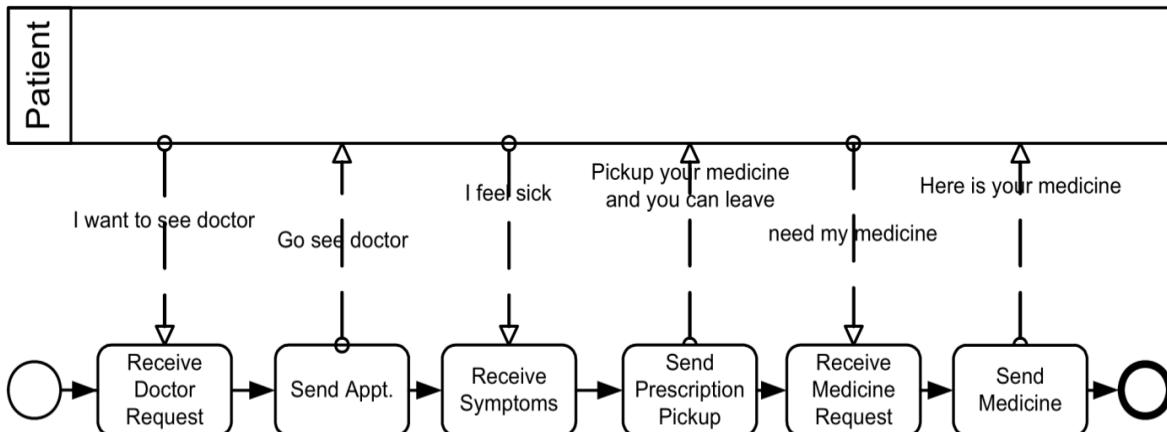
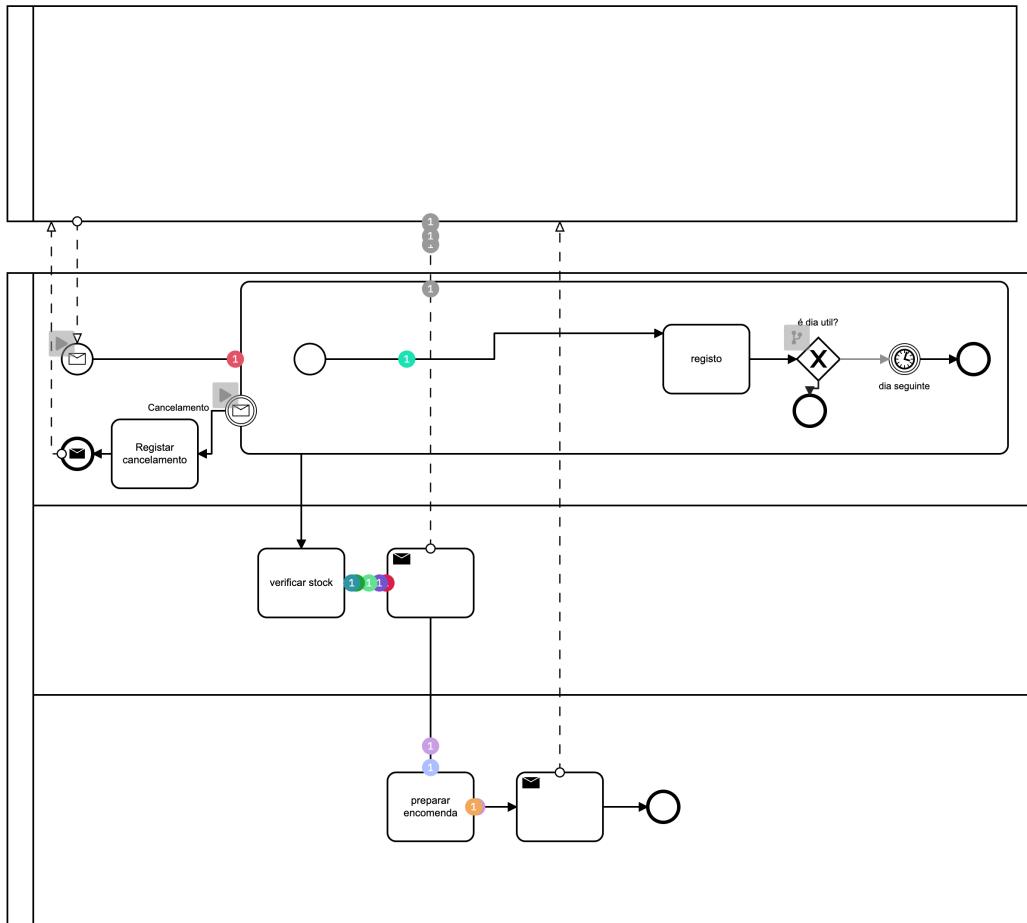


Figure 7.2 - Example of a *public* Process



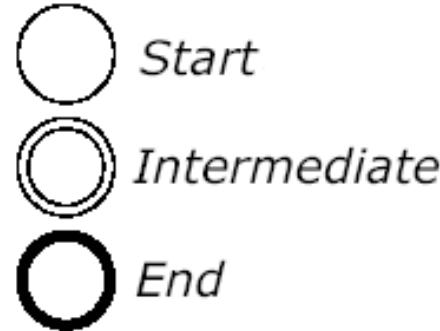
Process Definition vs. Process Instance

Flow Objects

- **Activity**: a unit of work.



- **Event**: an occurrence during a business process.



- **Gateway**: controls the flow of activities.



Connectors

Sequence Flow



defines the execution order of activities.

Default Flow

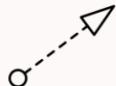


is the default branch to be chosen if all other conditions evaluate to false.

Conditional Flow



has a condition assigned that defines whether or not the flow is used.



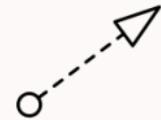
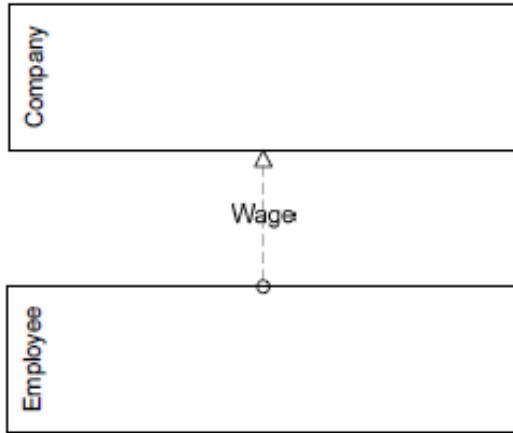
Message Flow

symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.



An Association is used to link information and Artifacts with BPMN graphical elements (see page 67). Text Annotations (see page 71) and other Artifacts (see page 66) can be Associated with the graphical elements. An arrowhead on the Association indicates a direction of flow (e.g., data), when appropriate.

Message Flow



Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.

Activities



A **Task** is a unit of work, the job to be performed. When marked with a  symbol it indicates a **Sub-Process**, an activity that can be refined.

Activity Markers

Markers indicate execution behavior of activities:



Sub-Process Marker



Loop Marker



Parallel MI Marker



Sequential MI Marker



Ad Hoc Marker



Compensation Marker

Task Types

Types specify the nature of the action to be performed:



Send Task



Receive Task



User Task



Manual Task

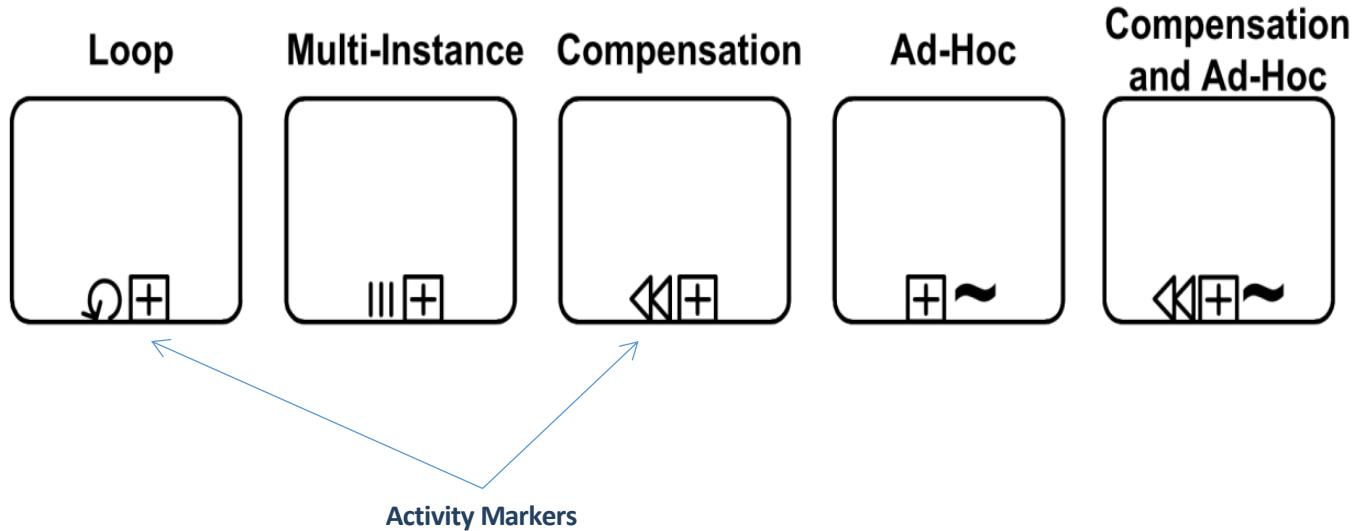


Business Rule Task

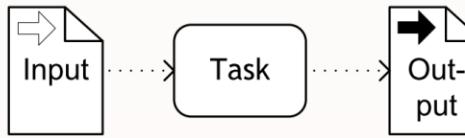


Service Task

Activity Types



Data



A **Data Input** is an external input for the entire process. It can be read by an activity.

A **Data Output** is a variable available as result of the entire process.



A **Data Object** represents information flowing through the process, such as business documents, e-mails, or letters.



A **Collection Data Object** represents a collection of information, e.g., a list of order items.

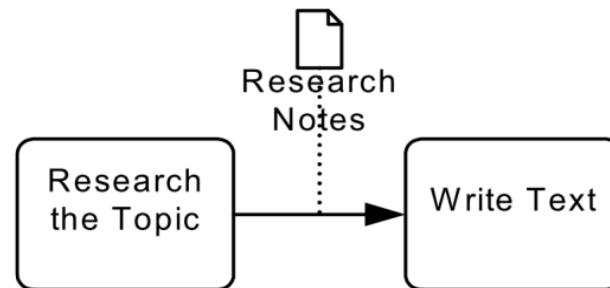
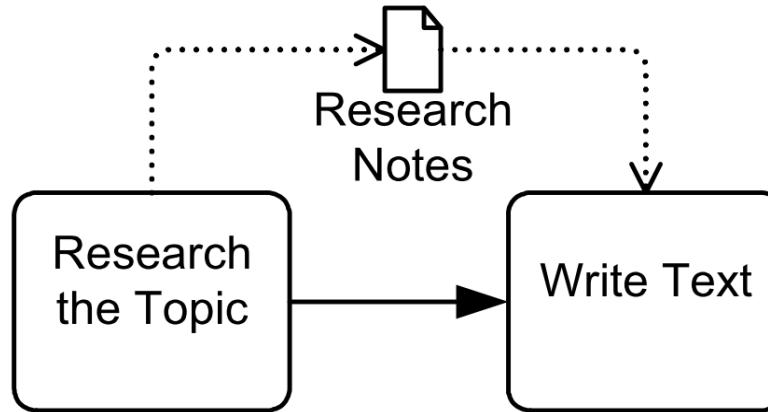


A **Data Store** is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.



A **Message** is used to depict the contents of a communication between two Participants.

Data Association within a Process



Event Triggers

- A **Trigger** specifies what causes the Event.
- The **Trigger** is shown as an icon inside the Event symbol.
- BPMN defines several types of **Triggers**:
 - None (i.e. no Trigger) and
 - 12 other (Message, Timer, Error, Signal,...)
- A **Trigger** may have two different **behaviours**:

Throw a Trigger



(throw/send a message)

Catch a Trigger



(catch/receive a message)

Event Behaviour (throw/catch)

- When **throwing** a Trigger the Event **waits for a token** and then **produces the Trigger**.
The notation for throw is a *filled* Trigger.
- When **catching** a Trigger the Event **waits for a token** and then **waits for the Trigger**.
The notation for catch is an *outlined* Trigger.
“Catch” is **blocking** while it waits for the Trigger and the Token.



- Start events:** can only **catch**
- Intermediate events:** can **catch** and **throw**
- End events:** can only **throw**



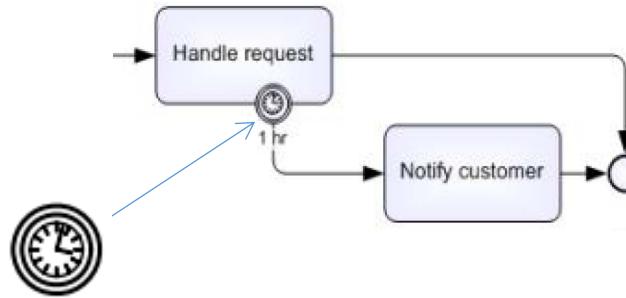
Interrupting and Non-Interrupting Events

Interrupting



Interrupting: when the event is thrown or caught the corresponding activity is interrupted and is not completed.

Notation is a **solid line**.

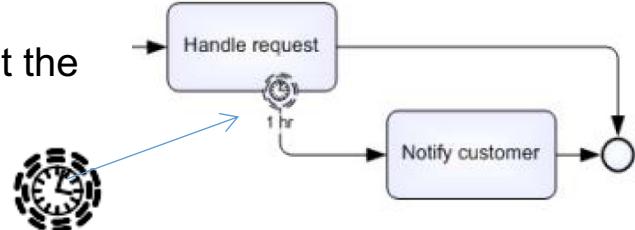


Non-Interrupting



Non-Interrupting: when the event is thrown or caught the corresponding activity continues its execution.

Notation is a **dotted line**.



Events

Events

Important Triggers

None: Untyped events, indicate start point, state changes or final states.

Message: Receiving and sending messages.

Timer: Cyclic timer events, points in time, time spans or timeouts.

Escalation: Escalating to an higher level of responsibility.

Conditional: Reacting to changed business conditions or integrating business rules.

Link: Off-page connectors. Two corresponding link events equal a sequence flow.

Error: Catching or throwing named errors.

Cancel: Reacting to cancelled transactions or triggering cancellation.

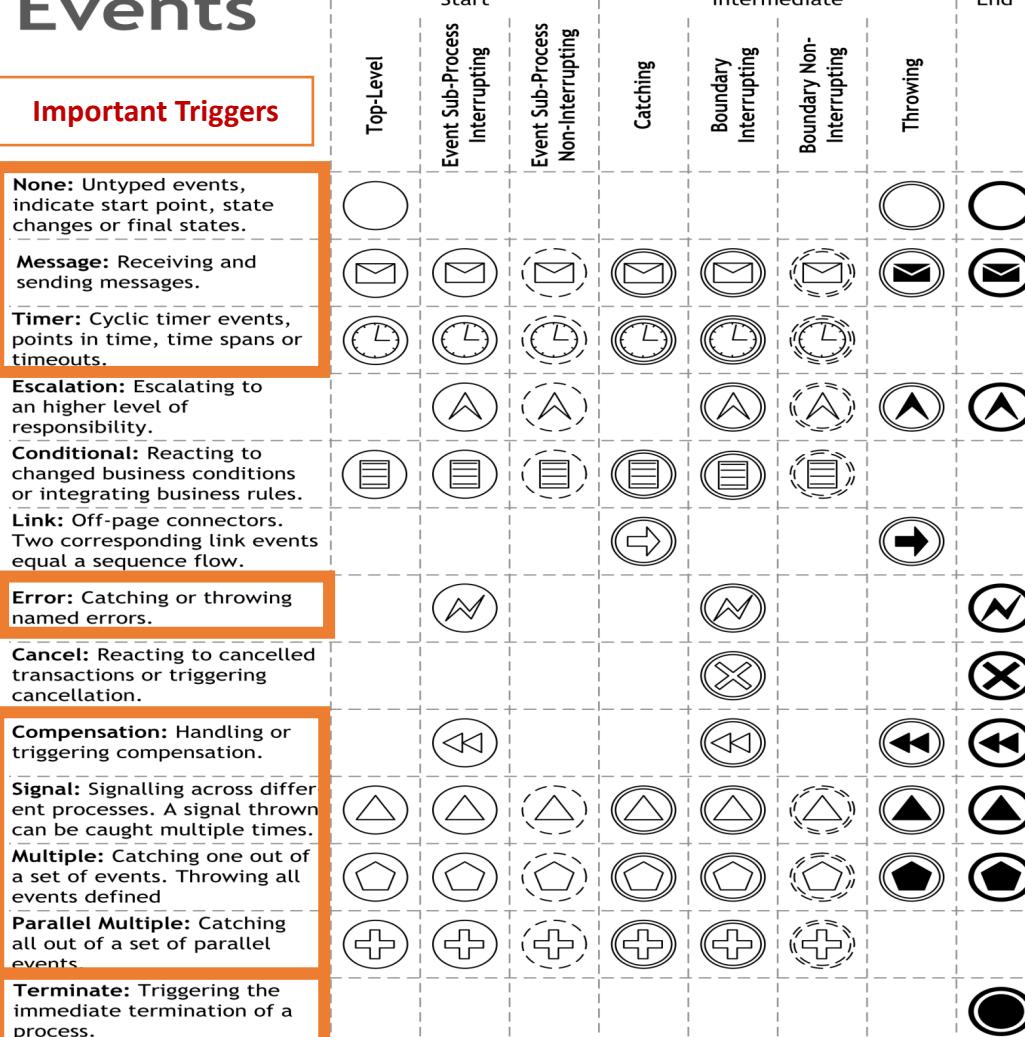
Compensation: Handling or triggering compensation.

Signal: Signalling across different processes. A signal thrown can be caught multiple times.

Multiple: Catching one out of a set of events. Throwing all events defined

Parallel Multiple: Catching all out of a set of parallel events

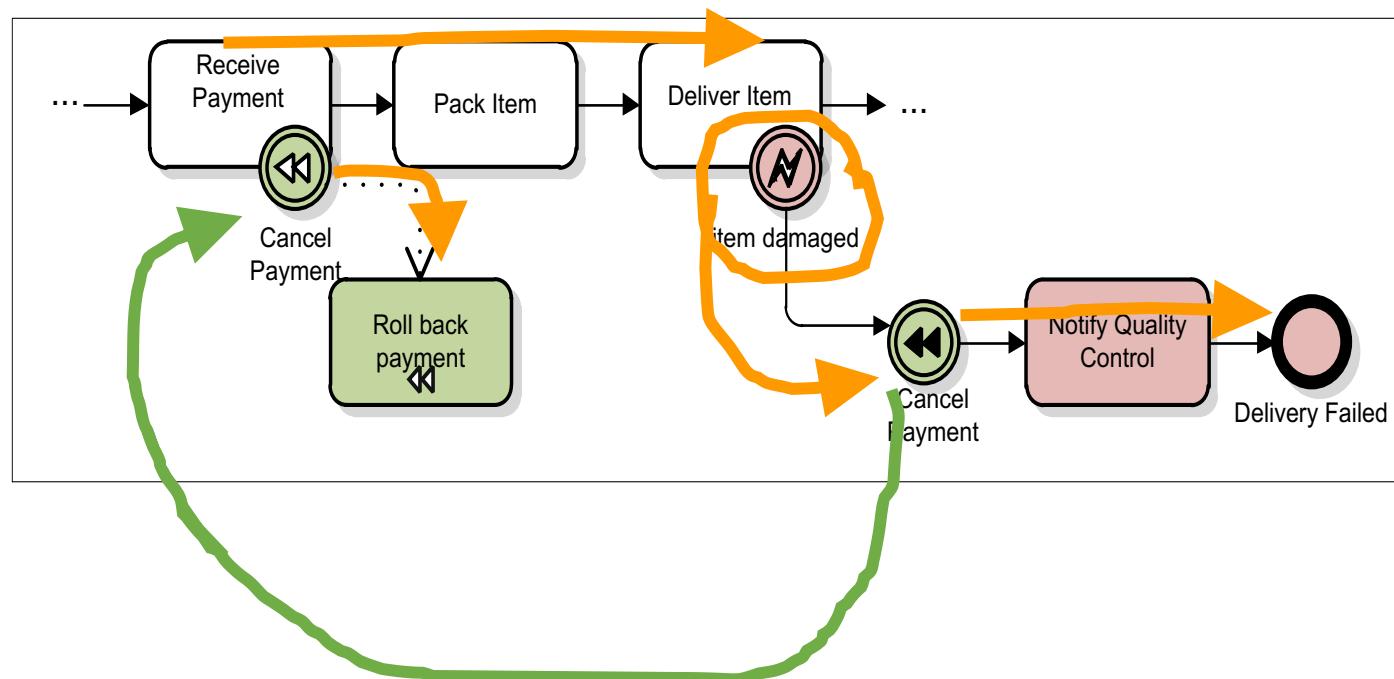
Terminate: Triggering the immediate termination of a process.



Compensation

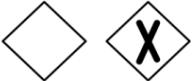
- Sometimes you need to undo the effects of a business process. You do this by **rolling it backwards**, one completed Activity at a time
- There are three options for undoing each completed Activity:
 1. **Do nothing** - there are no data changes to undo
 2. **Overwrite** - restore all data to its original state
 3. **Undo** - perform a specific Activity to undo the data changes - this is known as **compensation**

Errors and Compensation



Gateways

Exclusive Gateway



When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

Event-based Gateway



Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

Parallel Gateway



When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.



Inclusive Gateway

When splitting, one or more branches are activated. All active incoming branches must complete before merging.



Exclusive Event-based Gateway (instantiate)

Each occurrence of a subsequent event starts a new process instance.



Complex Gateway

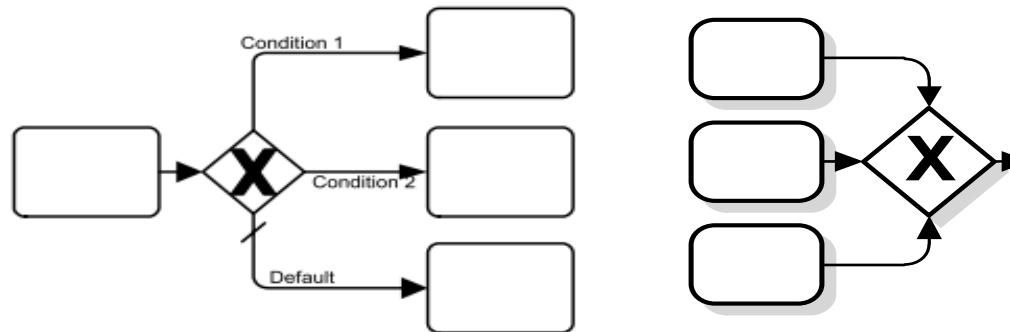
Complex merging and branching behavior that is not captured by other gateways.



Parallel Event-based Gateway (instantiate)

The occurrence of all subsequent events starts a new process instance.

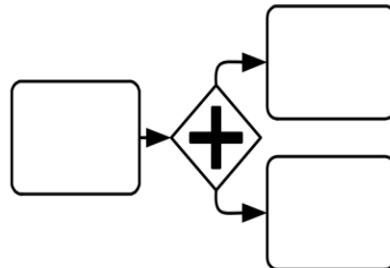
Exclusive Gateway (XOR-split/merge)



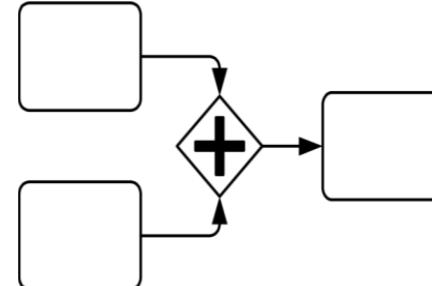
- **XOR-split** selects one and only one of the 2..* output flows.
- The conditions are evaluated top-to-bottom; the output flow associated to first condition evaluated true is selected.
- Optionally, a *default* flow (--) may be included. The default flow is unconditional and is selected whenever all the other conditions are evaluated false.
- The output flows of a XOR-split may be merged using a XOR-merge that continues as soon as one input flow arrives.

Parallel (AND-split/merge)

- AND-split forks one input flow into 2..* parallel (concurrent in time) output flows.
- AND-merge waits for *all* input flows before joining them into a single output flow.



AND-split (fork)



AND-merge(join)

Event-Based Gateway

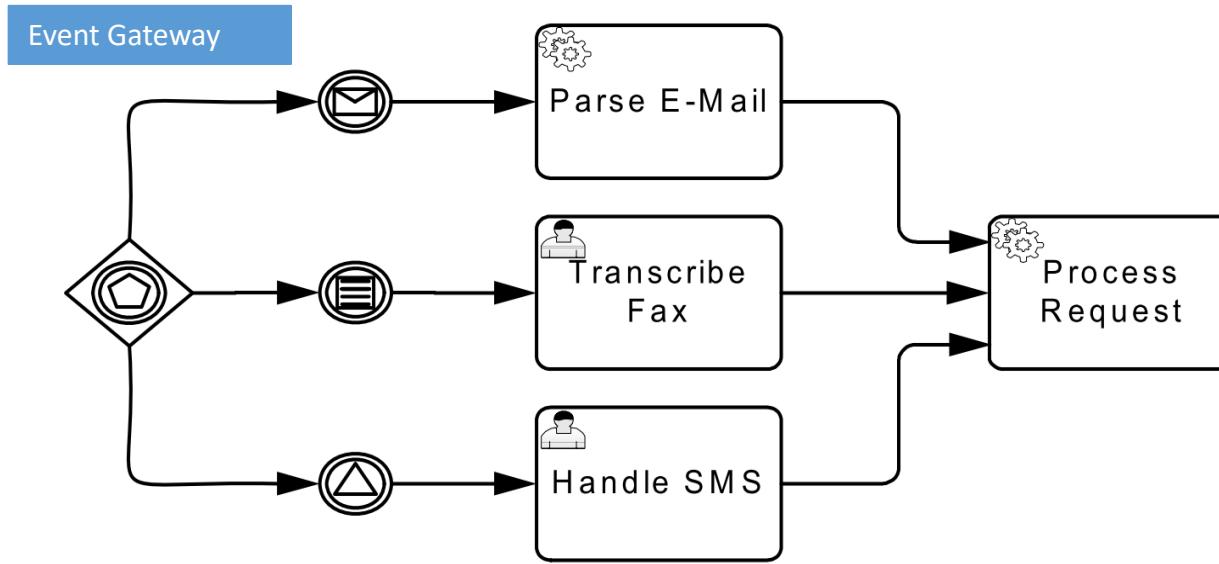
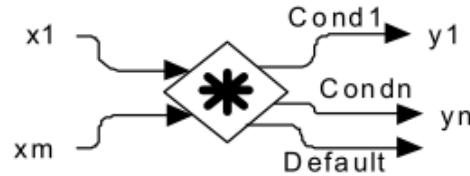


Figure 10.98 - A Process initiated by an Event-Based Gateway

Complex Gateway

- These have no default semantics!
 - Splitting - modeller provides an IncomingCondition
 - Merging - modeller provides an OutgoingCondition
- They *must* be supported by Text Annotations that describe their semantics, otherwise the BPD is unreadable!



```
cond1 := x2 < 5
cond2 := x1 > 10 and x2 = 0
...
...
```

The **Complex Gateway** can be used to model complex synchronization behavior. An Expression activationCondition is used to describe the precise behavior. For example, this Expression could specify that tokens on three out of five *incoming Sequence Flows* are needed to activate the **Gateway**. What tokens are produced by the **Gateway** is determined by conditions on the *outgoing Sequence Flows* as in the split behavior of the **Inclusive Gateway**. If tokens arrive later on the two remaining **Sequence Flows**, those tokens cause a reset of the **Gateway** and new token can be produced on the *outgoing Sequence Flows*. To determine whether it needs to wait for additional tokens before it can reset, the **Gateway** uses the synchronization semantics of the **Inclusive Gateway**.

Executable Business Process

Modelling an executable BPMN



Figure 1-2. A very simple process, that can already handle many requirements in the credit card example

As shown in Figure 1-3 the workflow engine:

- keeps the state persistent,
- schedules the retries,
- gives you visibility into current process instances and historic information.

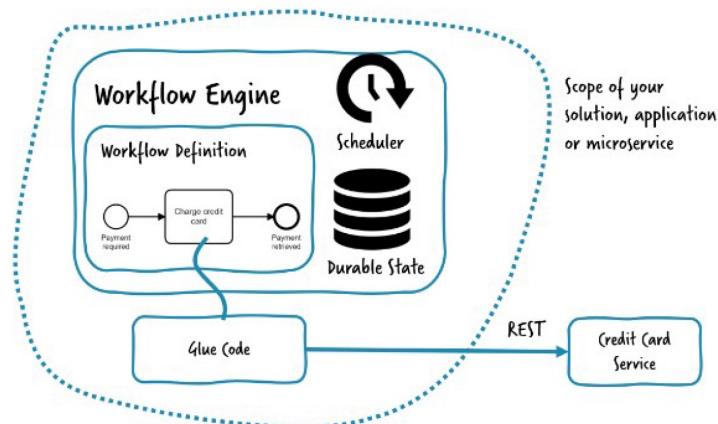


Figure 1-3. Workflow engine

Modelling an executable BPMN

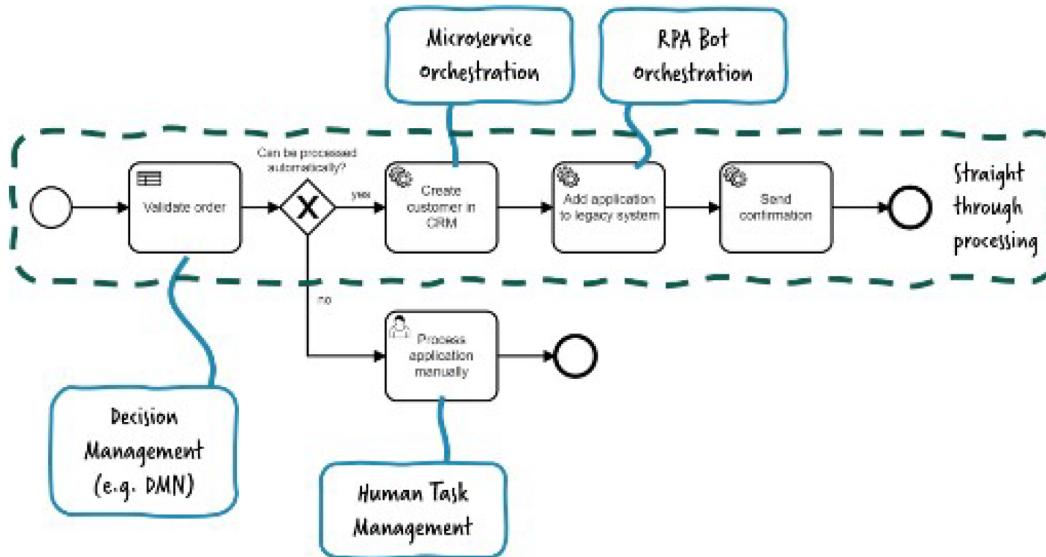


Figure 1-5. Use cases are typically mixed in real-life examples

Modelling an executable BPMN

Process automation is about automating tasks within a process as well as the automation of the control flow between these tasks.

Three different combinations possible which define a kind of maturity levels of process automation:

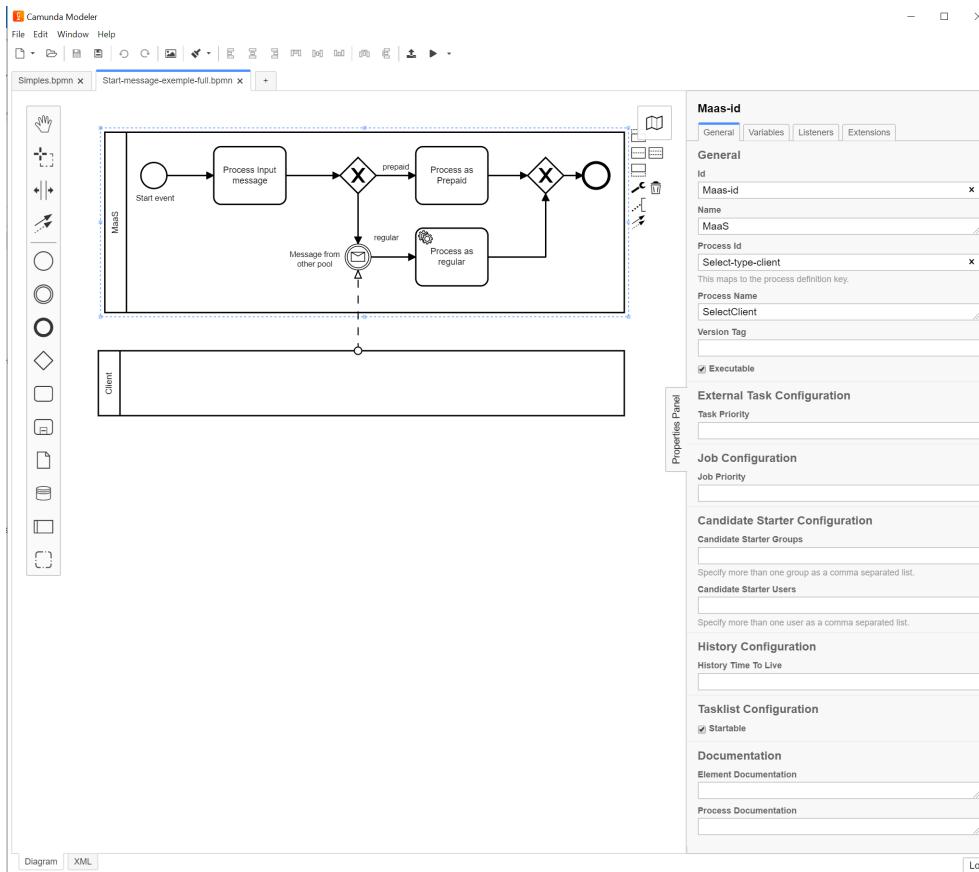
1. The **human controls** the process, often passing paper around. You're probably familiar with this from physical inbox folders on your desk in an office environment. Software and office tools can, of course, make this process more efficient.
2. The **computer controls** the process and involves people whenever necessary, for example using task list user interfaces.
3. The **whole process** is fully automated and only requires manual intervention if something happens beyond the expected normal operations.

Modelling an executable BPMN

Two types of automation:

1. **Automation of the control flow:** The interactions between tasks are automated, but the activities itself might not. In the previous example, this was the humans cooking the food. A process might not be fully automated as it cannot run without human interaction. This is often known as human task management.
2. **Automation of the tasks:** The task itself is automated. In the previous example, this would be the robots who cook the food. Automating the tasks leads to fully automated processes, which is typically named STP (straight through processing). Very often this is the most ideal path, where everything runs smoothly and is fully automated. Humans are involved in case any unforeseen exceptions or special cases occur. This approach is actually quite powerful, as you can invest the effort on automation on the majority of cases and save additional (manual) effort only on the rare cases.

Modelling an executable BPMN

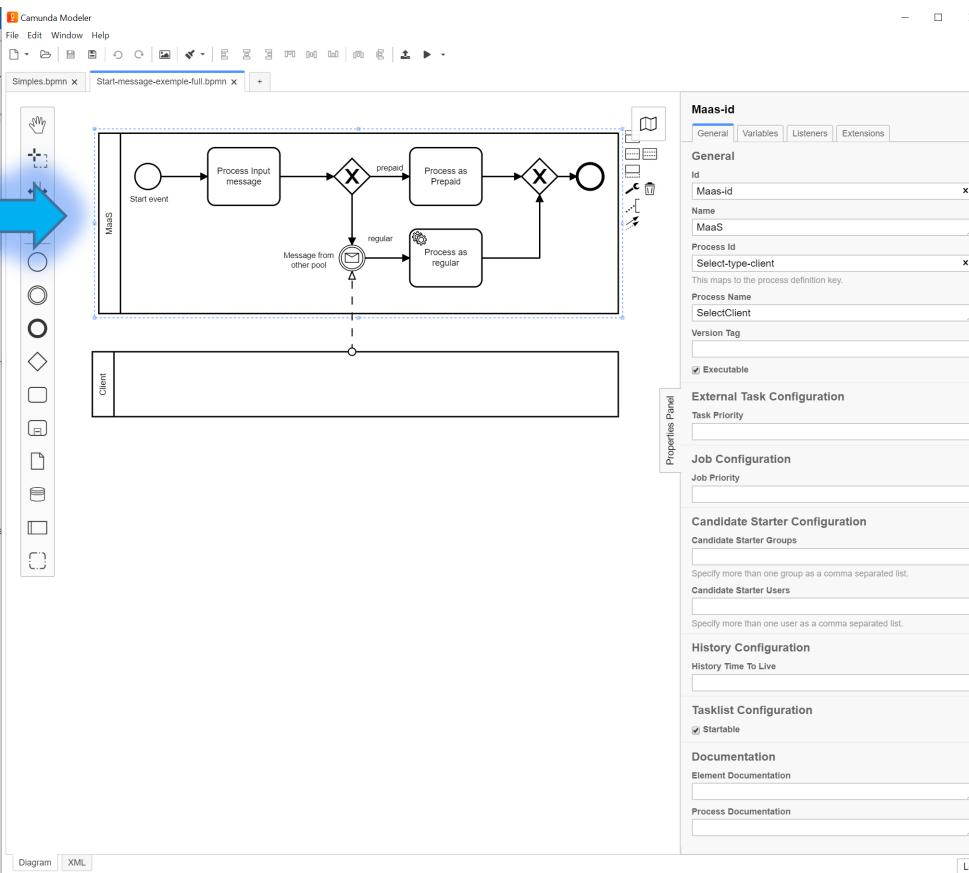


Modelling an executable BPMN

– Separation of systems

Each system should be modelled as a separate pool

Then, each pool uses communication between them sending and receiving messages



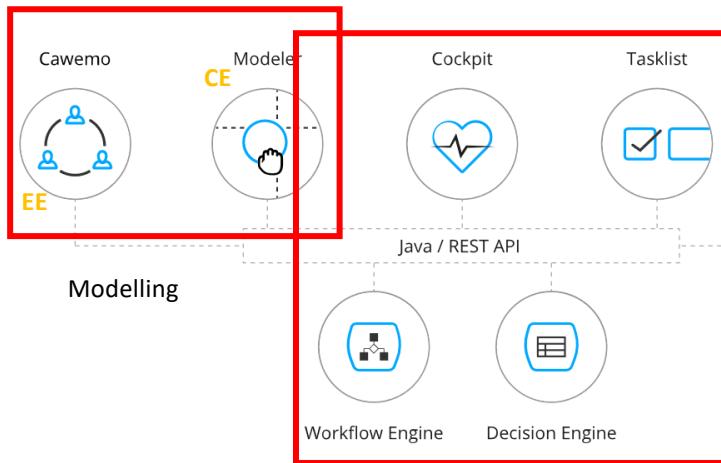
Later, in terms of execution, a process engine supports many systems or each system a different engine

A Business Process Execution engine

The case of Camunda

Camunda Architecture

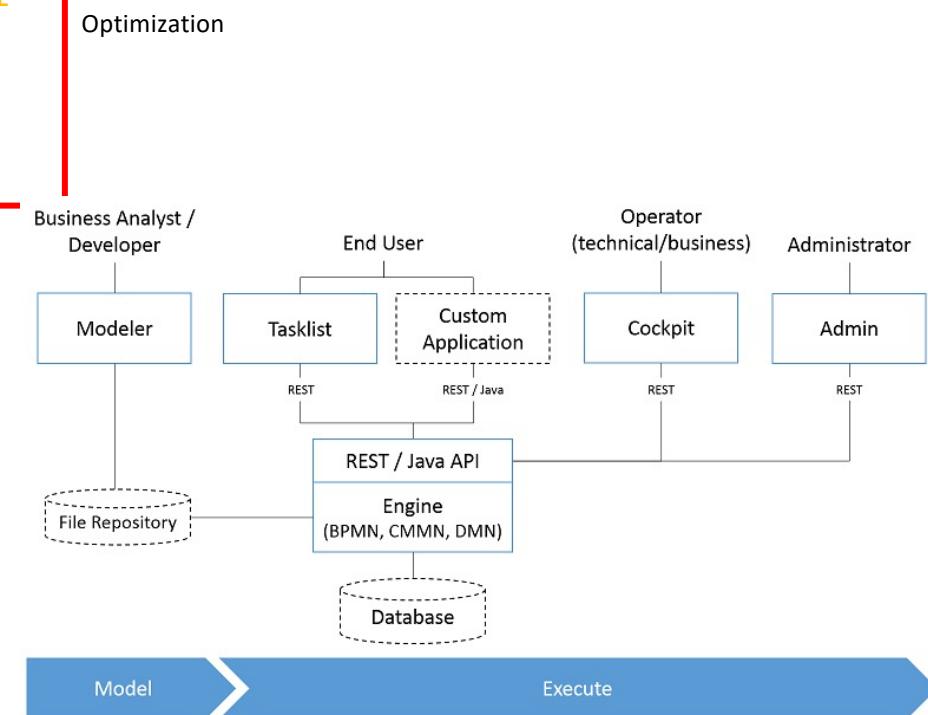
Camunda Components



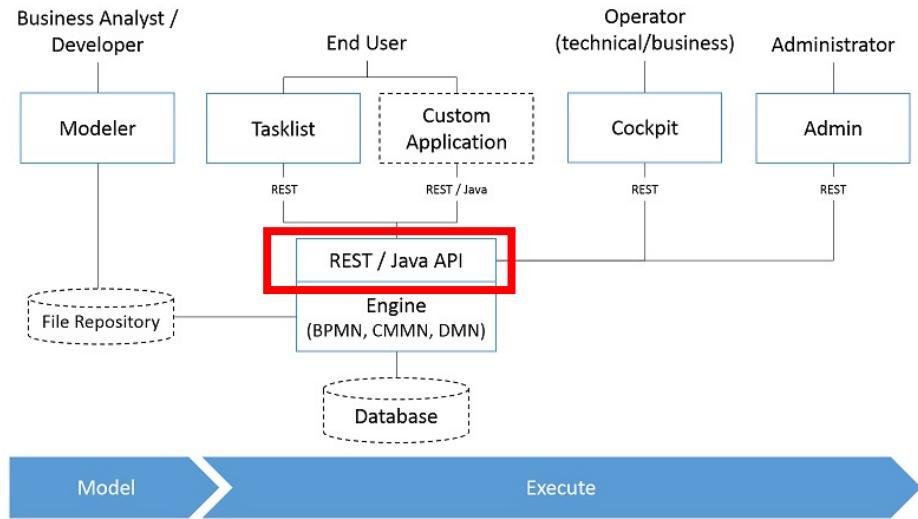
2 edições para optar:

CE - Community edition

EE - Enterprise edition



The Camunda REST API is provided by an OpenAPI description



Process engine functionalities are exposed by REST API and thus can be used by other applications

Camunda Architecture

https://docs.camunda.org/manual/latest/reference/rest/openapi/

Camunda Docs Get Started BPM Platform Optimize Camemo Enterprise Security

BPM Platform: latest (7.1.0)

- Introduction
- User Guide
- Reference
 - Rest API
 - Overview
 - OpenAPI**
 - Authorization
 - Batch
 - Case Definition
 - Case Execution
 - Case Instance
 - Condition
 - Decision Definition
 - Decision Requirements Definition
 - Deployment
 - Engine
 - Event Subscription
 - Execution
 - External Task
 - Filter
 - Group
 - History
 - Identity
 - Incident
 - Job
 - Job Definition
 - Message
 - Metrics
 - Migration
 - Modification
 - Process Definition
 - Process Instance
 - Signal
 - Schema Log
 - Task
 - Telemetry
 - Tenant
 - User
 - Variable Instance
 - Version
 - Javadoc:
 - BPMMN 2.0
 - DMN 1.3
 - CMIM 1.1
 - Forms
 - Spin Dataformats
 - Connectors
 - Deployment Descriptors
 - Installation

OPTIONS

ON THIS PAGE:

- Client Generation
- Getting Started Experience
- Coverage

Get up to 24/7 support with the Camunda Enterprise Edition

OpenAPI

The Camunda REST API has an OpenAPI description that follows the [OpenAPI Specification 3.0.2](#). OpenAPI is a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection.

The OpenAPI description brings options for:

- Client Generation in different languages: Providing flexibility for adoption of the process engine in many languages and possibility for custom implementation built on top.
- Getting Started Experience: Improving the getting started experience for the users with the option to try out the REST API following along with the documentation and examples.

The documentation is shipped as a single `openapi.json` file archived in a jar artifact. Download the Camunda REST API artifact containing the OpenAPI documentation [here](#). Choose the correct version and then download the jar file.

Alternatively, you can obtain this artifact with the following Maven coordinates:

```
<dependency>
<groupId>org.camunda.bpm</groupId>
<artifactId>camunda-engine-rest-openapi</artifactId>
<version>${version.camunda}</version>
</dependency>
```

Client Generation

To generate REST API client in the language of your preference based on the OpenAPI documentation, you will need a client generator library, e.g. [OpenAPI Generator](#) or any other library that is compatible with OpenAPI Specification version 3.0.

Follow the steps of the OpenAPI Generator's documentation, how to [install](#) the tool and [generate a simple client](#) in one of the supported languages.

Getting Started Experience

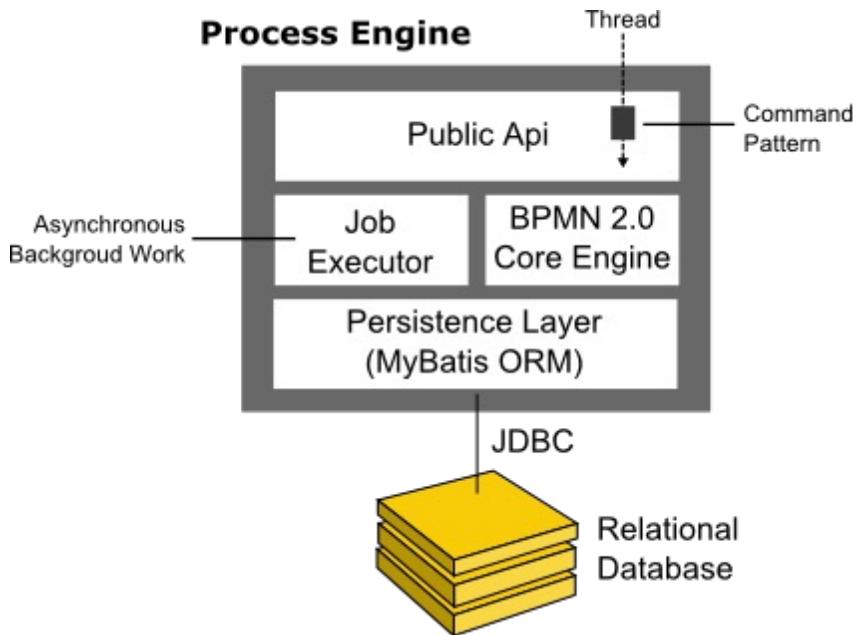
Instead of client generation, you can use one of the OpenAPI editors to play around with the REST API endpoints.

For example, go to [Swagger Editor](#) and paste the content of the `openapi.json` on the left-hand side in the editor. Start a Process engine with enabled cross-origin requests, and you will be able to execute requests from the editor.

Some API Tools also support import of endpoints via upload of a OpenAPI document. For example, [Postman](#) users can [import the OpenAPI documentation](#) and work with the REST endpoints from a single place.

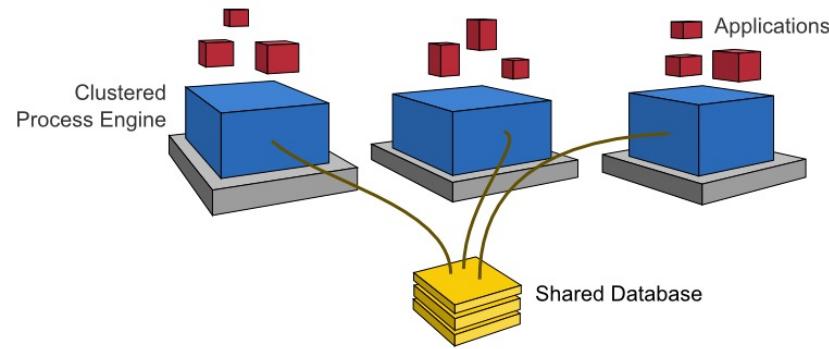
<https://docs.camunda.org/manual/latest/reference/rest/openapi/>

Camunda Architecture

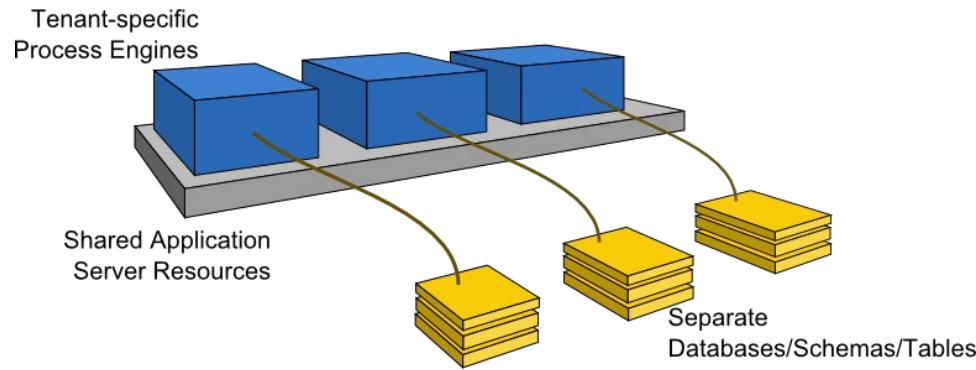


- **Process Engine Public API:** Service-oriented API allowing Java applications to interact with the process engine. The different responsibilities of the process engine (i.e., Process Repository, Runtime Process Interaction, Task Management, ...) are separated into individual services. The public API features a [command-style access pattern](#):
- **BPMN 2.0 Core Engine:** this is the core of the process engine. It features a lightweight execution engine for graph structures (PVM - Process Virtual Machine), a BPMN 2.0 parser which transforms BPMN 2.0 XML files into Java Objects and a set of BPMN Behavior implementations (providing the implementation for BPMN 2.0 constructs such as Gateways or Service Tasks).
- **Job Executor:** the Job Executor is responsible for processing asynchronous background work such as Timers or asynchronous continuations in a process.
- **The Persistence Layer:** the process engine features a persistence layer responsible for persisting process instance state to a relational database.

Camunda Deployment Models

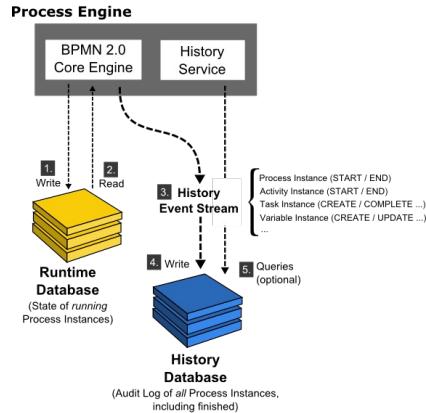
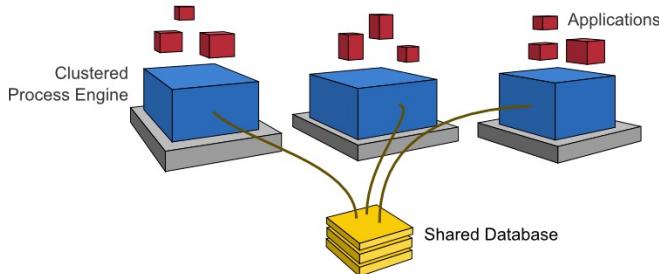


Clustering



Multi-tenancy

Camunda Deployment & Operation environment



- **JAVA compatibility:**
 - Oracle JDK 8 / 9 / 10 / 11 / 12 / 13 / 14 / 15
 - IBM JDK 8 (with J9 JVM)
 - OpenJDK 8 / 9 / 10 / 11 / 12 / 13 / 14 / 15, including builds of the following products: Oracle OpenJDK, AdoptOpenJDK (with HotSpot JVM), Amazon Corretto, Azul Zulu
- **Process Engine, Camunda cockpit, Tasklist e Admin compatibility:**
 - Apache Tomcat 9.0,
 - JBoss EAP 7.0 / 7.1 / 7.2 / 7.3 / 7.4,
 - Wildfly Application Server 13.0 / 14.0 / 15.0 / 16.0 / 17.0 / 18.0 / 19.0 / 20.0 / 21.0 / 22.0 / 23.0,
 - IBM WebSphere Application Server 8.5 / 9.0 ([Enterprise Edition only](#))
 - Oracle WebLogic Server 12c (12R2) ([Enterprise Edition only](#))
- **Shared database compatibility:**
 - MySQL 5.7 / 8.0
 - MariaDB 10.2 / 10.3
 - Oracle 12c / 19c
 - IBM DB2 11.1 (excluding IBM z/OS for all versions)
 - PostgreSQL 9.6 / 10 / 11 / 12 / 13
 - Amazon Aurora PostgreSQL compatible with PostgreSQL 9.6 / 10.4 / 10.7 / 10.13 / 12.4
 - Microsoft SQL Server 2014/2016/2017/2019 (see [Configuration Note](#))
 - Microsoft Azure SQL with Camunda-supported SQL Server compatibility levels (see [Configuration Note](#)):
 - SQL Server on Azure Virtual Machines
 - Azure SQL Managed Instance
 - Azure SQL Database
 - H2 1.4 (not recommended for [Cluster Mode](#) - see [Deployment Note](#))
 - CockroachDB v20.1.3 (see [Configuration guide](#) for more details)
- **Clustering** – active/active, with shared database for process execution
- **Kubernetes** for dynamic installation
- **Separation** between execution data and historical data
- **Optimize** available with Enterprise Edition

Camunda 8 connectors

[Home](#) > [Connectors](#) > [Out-of-the-box Connectors](#) > [Overview](#)

Version: 8.1

Overview

This section gives an overview of the **out-of-the-box Connectors** available in Camunda Platform 8. All Connectors are available for Camunda Platform 8 SaaS and **Self Managed**.

Outbound Connectors

- [Amazon SNS Connector](#) - Send messages to [Amazon Simple Notification Service](#) from your BPMN process.
- [Amazon SQS Connector](#) - Send messages to [Amazon Simple Queue Service](#) from your BPMN process.
- [AWS Lambda Connector](#) - Invoke [AWS Lambda Functions](#) from your BPMN process.
- [Google Drive Connector](#) - Create folders or files from a [Google Drive](#) template from your BPMN process.
- [GraphQL Connector](#) - Execute a [GraphQL](#) query or mutation from your BPMN process.
- [Kafka Producer Connector](#) - Produce messages to [Kafka](#) from your BPMN process.
- [Microsoft Teams Connector](#) - Interactions with [Microsoft Teams](#) from your BPMN process.
- [RabbitMQ Connector](#) - Send messages to [RabbitMQ](#) from your BPMN process.
- [REST Connector](#) - Make a request to a REST API and use the response in the next steps of your process.
- [SendGrid Connector](#) - Quickly send emails from your BPMN processes.
- [Slack Connector](#) - Send messages to channels or users in your [Slack](#) workspace from your BPMN process.
- [UiPath Connector](#) - Orchestrate your [UiPath](#) Bots with Camunda.

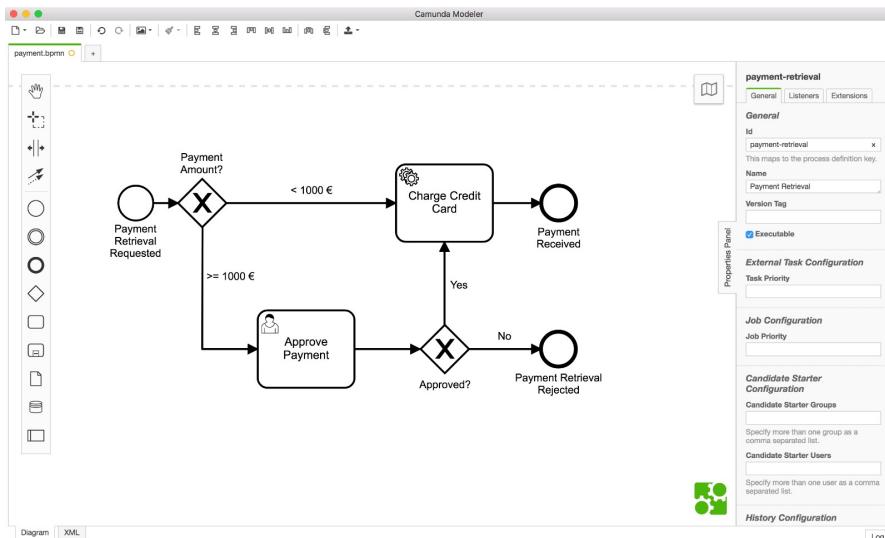
Inbound Connectors

- [HTTP Webhook Connector](#) - Start a process instance with your custom webhook configuration.
- [GitHub Webhook Connector](#) - Start a process instance triggered by a [GitHub](#) event.

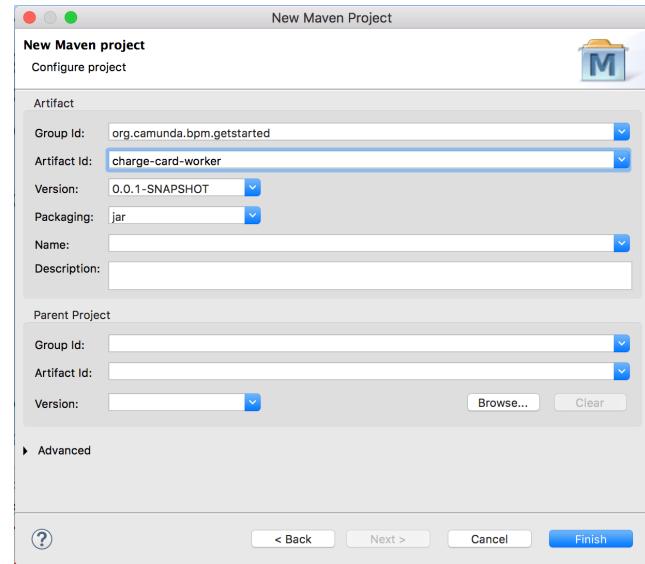
In addition to this section on Connectors, we recommend reviewing [Connector secrets](#).

If you want to build **custom Connectors**, head over to our [Connector SDK guide](#).

The development environment is based on



Camunda Modeler – this will be the option for Enterprise Integration course

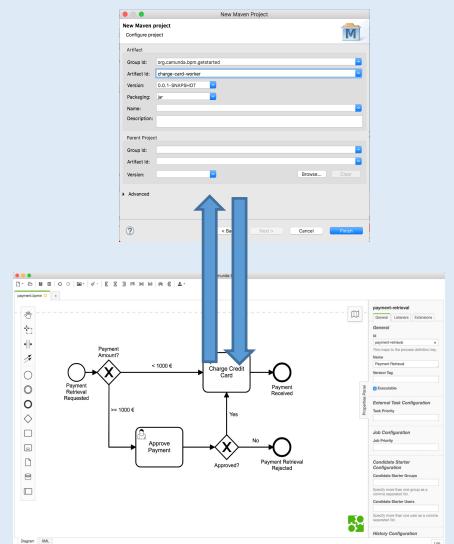


Eclipse with Maven for JAVA classes

Two architectural models for tasks' execution

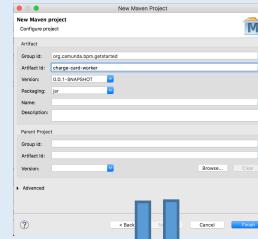
PUSH

JVM

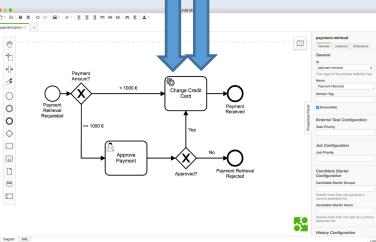


PULL

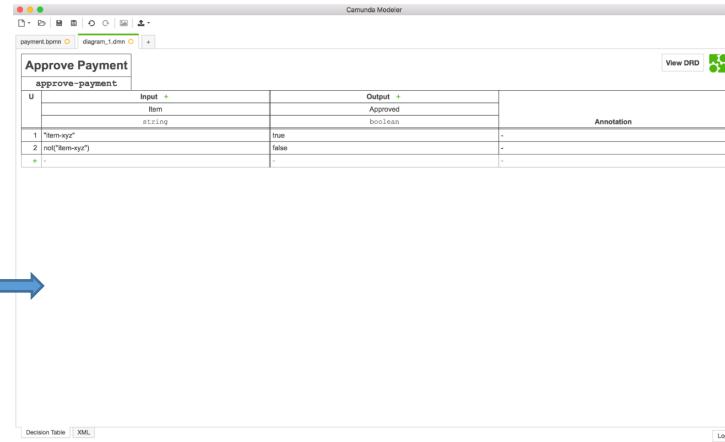
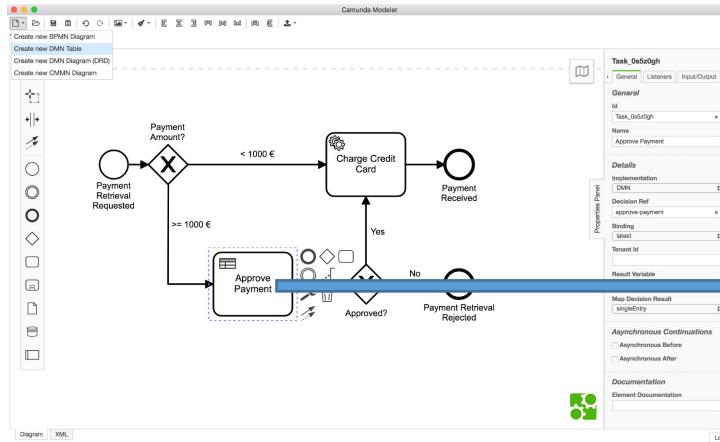
JVM



JVM

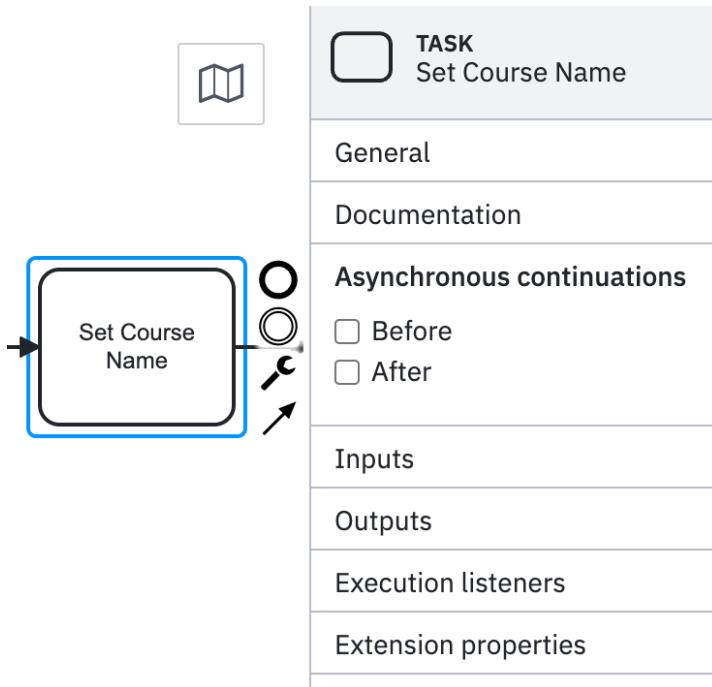


Business rules specification



Camunda implements the OMG's DMN specification, making it easy to model multiple Gateways into simpler-to-manage rule tables.

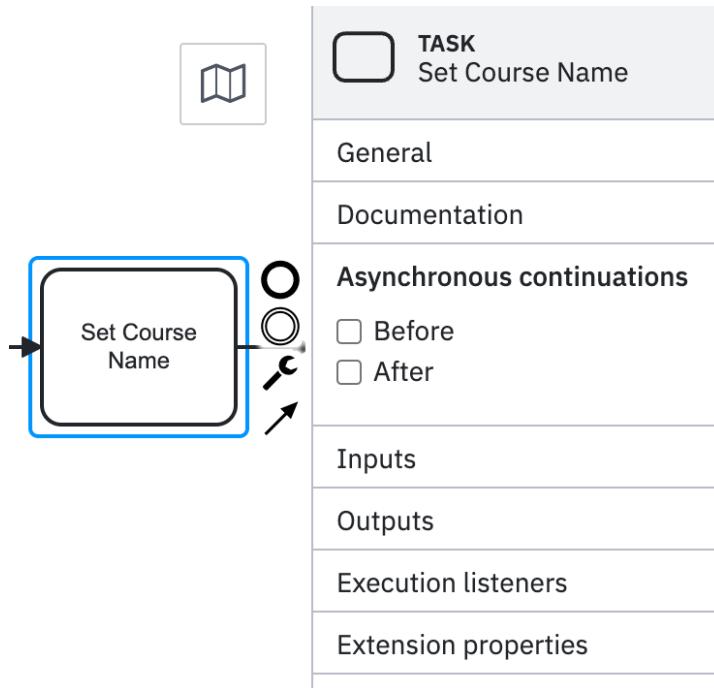
Asynchronous Continuations in Camunda BPM



Asynchronous continuations are break-points in the process execution. They are used as transaction boundaries and allow another thread than the currently active thread to continue execution.

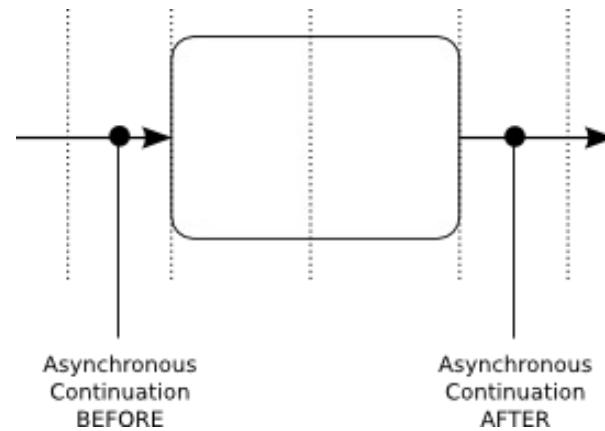
- Async is used for placing a safe-point before an activity such that the execution state is committed. If the activity then fails to execute, the transaction is rolled back only up to the safe point.
- Async also comes in handy if you have longer-running computations and do not want to block the calling thread (eg. HTTP Thread) but instead want to delegate the heavy lifting to a background thread.
- Finally, due to the fact that asynchronous continuations are executed by the job executor, the retry mechanism can be used in order to retry a failed activity execution.

Asynchronous Continuations in Camunda BPM



The screenshot shows the configuration interface for a task named "Set Course Name". On the left, there's a large blue-bordered box containing the task name and a small icon. To its right is a vertical list of configuration sections:

- General
- Documentation
- Asynchronous continuations**
 - Before
 - After
- Inputs
- Outputs
- Execution listeners
- Extension properties



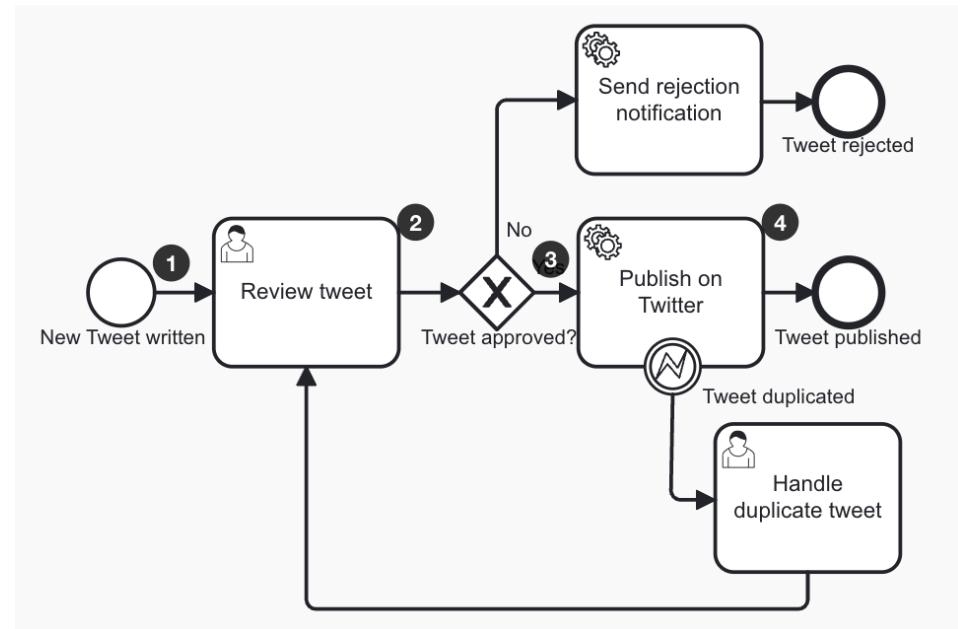
Handling data in processes

When using Camunda, you have access to a dynamic map of process variables, which lets you associate data to every single process instance (and local scopes in case of user tasks or parallel flows).

- 1 The process instance starts with a freshly written tweet we need to remember.
- 2 We need to present this tweet so that the user can decide whether to approve it.
- 3 The gateway needs to have access to this information: was the tweet approved?
- 4 To publish the tweet, the service task again needs the tweet itself!

Therefore, the tweet approval process needs two variables:

Variable name	Variable type	Sample value
tweet	String	"@Camunda rocks"
approved	Boolean	true



Input and output variables

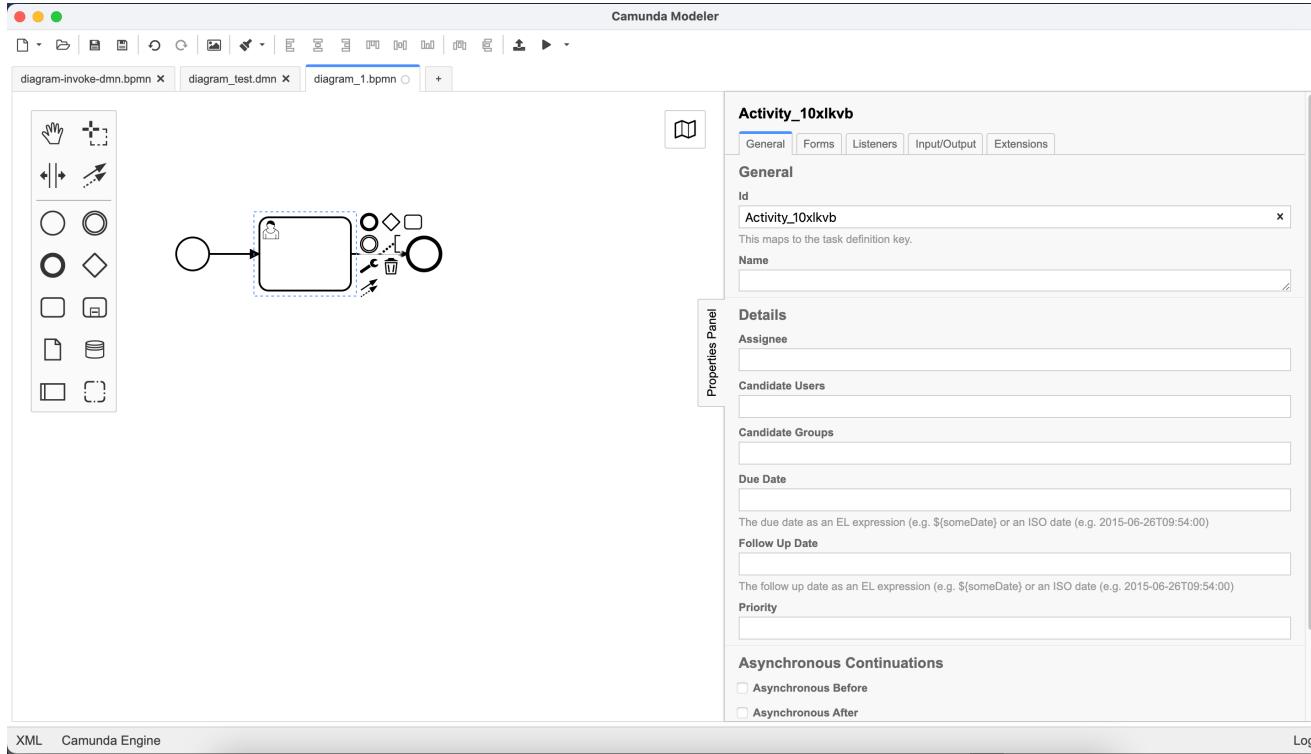
- Input mappings can be used to create new variables. They can be defined on service tasks and subprocesses. When an input mapping is applied, it creates a new **local variable** in the scope where the mapping is defined.
- Output mappings can be used to customize how job/message variables are merged into the process instance. They can be defined on service tasks, receive tasks, message catch events, and subprocesses.
- Can be used to create new variables or customize how variables are merged into the process instance.

Process instance variables	Input mappings	New variables
orderId: "order-123"	source: <code>=orderId</code> target: <code>reference</code>	reference: "order-123"
customer:{name: "John"}	source: <code>=customer.name</code> target: <code>sender</code>	sender: "John"
customer: "John" iban: "DE456"	source: <code>=customer</code> target: <code>sender.name</code> source: <code>=iban</code> target: <code>sender.iban</code>	sender: {"name": "John", "iban": "DE456"}

Job/message variables	Output mappings	Process instance variables
status: "0k"	source: <code>=status</code> target: <code>paymentStatus</code>	paymentStatus: "0k"
result: {"status": "0k", "transactionId": "t-789"}	source: <code>=result.status</code> target: <code>paymentStatus</code> source: <code>=result.transactionId</code> target: <code>transactionId</code>	paymentStatus: "0k" transactionId: "t-789"
status: "0k" transactionId: "t-789"	source: <code>=transactionId</code> target: <code>order.transactionId</code>	order: {"transactionId": "t-789"}

Source	Target
customer.name	sender
customer.iban	iban
totalPrice	price
orderId	reference

Camunda Authorization

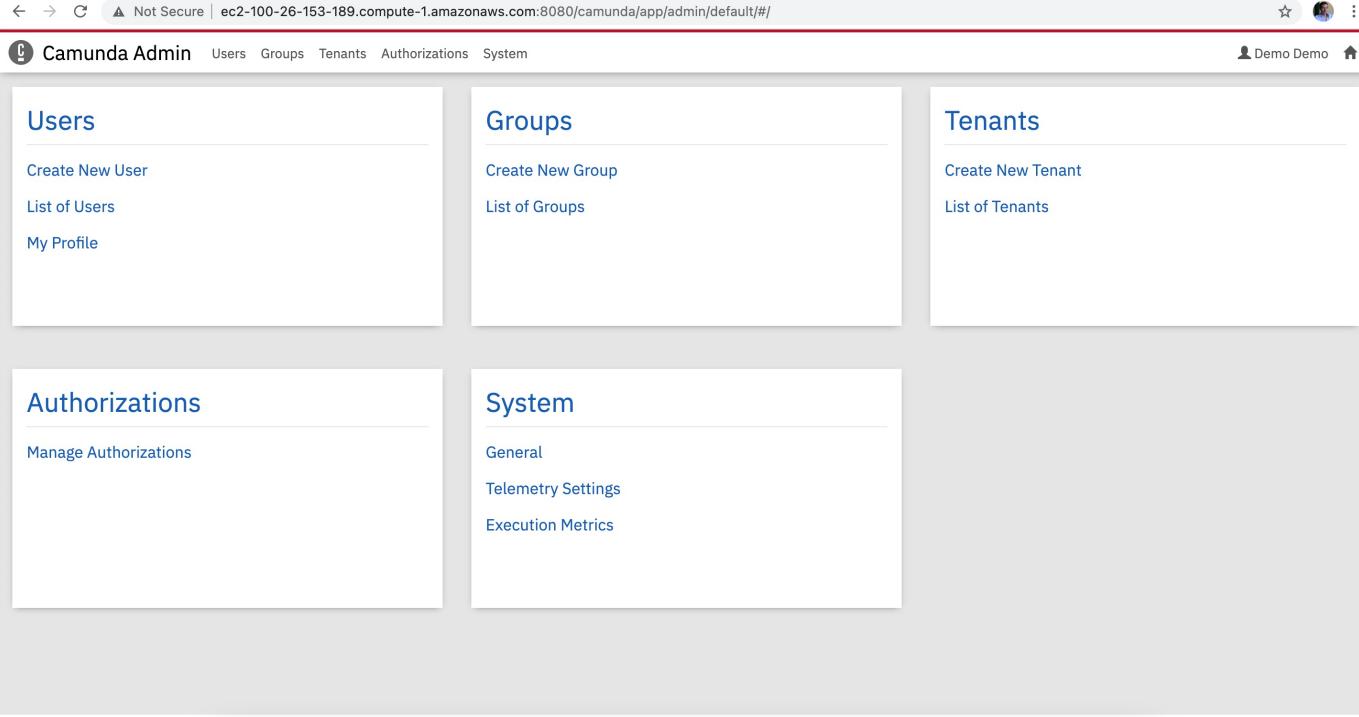


- **Assignee**
a specific user who must perform the task
- **Candidate users**
a list of specific users who can perform a task
- **Candidate groups**
a list of user groups who can perform the task

Fine-grained authorizations definition

<https://docs.camunda.org/manual/latest/user-guide/process-engine/authorization-service/>

<https://docs.camunda.org/optimize/2.2/technical-guide/authorization/>



The screenshot shows the Camunda Admin interface with the following navigation structure:

- Camunda Admin** (User icon) - Demo
- Users
- Groups
- Tenants
- Authorizations
- System

Under each section, there are links to create new entities or view lists:

- Users**: Create New User, List of Users, My Profile
- Groups**: Create New Group, List of Groups
- Tenants**: Create New Tenant, List of Tenants
- Authorizations**: Manage Authorizations
- System**: General, Telemetry Settings, Execution Metrics

At the bottom of the page, there is a footer note: "Date and Time displayed in local timezone: Europe/Lisbon" and "Powered by Camunda Platform / v7.15.0".

Creating a new group

Not Secure | ec2-100-26-153-189.compute-1.amazonaws.com:8080/camunda/app/admin/default/#/groups?searchQuery=%5B%5D

Camunda Admin Users Groups Tenants Authorizations System Demo Demo

Dashboard » Groups

List of groups Create new group +

Add criteria 5 ⚡ H ▾

ID ▲	Name ▬	Type ▬	Action
25	Sales	Organizational Unit	Edit
accounting	Accounting	WORKFLOW	Edit
camunda-admin	camunda BPM Administrators	SYSTEM	Edit
management	Management	WORKFLOW	Edit
sales	Sales	WORKFLOW	Edit

Success : ✘ Successfully created new group 25

Authorizing a group to define a process

← → ⌂ Not Secure | ec2-100-26-153-189.compute-1.amazonaws.com:8080/camunda/app/admin/default#/authorization?resource=6

 Camunda Admin Users Groups Tenants Authorizations System  Demo Demo 

Dashboard » Authorizations

Process Definition Authorizations				
Type	User / Group	Permissions	Resource ID	Action
ALLOW	# accounting	READ, READ_HISTORY	invoice	Edit Delete
ALLOW	# camunda-admin	ALL	*	Edit Delete
ALLOW	# management	READ, READ_HISTORY	invoice	Edit Delete
ALLOW	# sales	READ, READ_HISTORY	invoice	Edit Delete
ALLOW	<input type="text" value="User / Group ID"/>	ALL	<input type="text" value="Resource ID"/>	<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input checked="" type="checkbox"/> <input type="checkbox"/>

[Create new authorization +](#)

Application
 Authorization
 Batch
 Decision Definition
 Decision Requirements Definition
 Deployment
 Filter
 Group
 Group Membership
 Historic Process Instance
 Historic Task Instance
 Operation Log
Process Definition
 Process Instance
 Task
 Tenant
 Tenant Membership
 User

Success : ×
 Successfully created new group 25

Date and Time displayed in local timezone: Europe/Lisbon

Powered by [Camunda Platform](#) / v7.15.0

Authorizing a group to instantiate a process

Not Secure | ec2-100-26-153-189.compute-1.amazonaws.com:8080/camunda/app/admin/default/#/authorization?resource=8

Camunda Admin Users Groups Tenants Authorizations System Demo Demo :

Dashboard » Authorizations

Create new authorization +

Type	User / Group	Permissions	Resource ID	Action
ALLOW	camunda-admin	ALL	*	Edit Delete

Application Authorization Batch Decision Definition Decision Requirements Definition Deployment Filter Group Group Membership Historic Process Instance Historic Task Instance Operation Log Process Definition Process Instance Task Tenant Tenant Membership User

Success : Successfully created new group 25

Date and Time displayed in local timezone: Europe/Lisbon

Powered by Camunda Platform / v7.15.0

Authorizing a group to a specific task of a process

Camunda Admin [Users](#) [Groups](#) [Tenants](#) [Authorizations](#) [System](#)

[Dashboard](#) » Authorizations

Task Authorizations				
Type	User / Group	Permissions	Resource ID	Action
ALLOW	# accounting	READ, UPDATE	65f88b10-ac15-11eb-95c4-0242ac110002	Edit Delete
ALLOW	# accounting	READ, UPDATE	67204280-ac15-11eb-95c4-0242ac110002	Edit Delete
ALLOW	# accounting	READ, UPDATE	676699d9-ac15-11eb-95c4-0242ac110002	Edit Delete
ALLOW	# accounting	READ, UPDATE	67b3f6d8-ac15-11eb-95c4-0242ac110002	Edit Delete
ALLOW	# camunda-admin	ALL	*	Edit Delete
ALLOW	@ demo	READ, UPDATE	67518b0a-ac15-11eb-95c4-0242ac110002	Edit Delete
ALLOW	@ demo	READ, UPDATE	67d89632-ac15-11eb-95c4-0242ac110002	Edit Delete
ALLOW	@ mary	READ, UPDATE	*	Edit Delete
ALLOW	# sales	READ, UPDATE	65f88b10-ac15-11eb-95c4-0242ac110002	Edit Delete
ALLOW	# sales	READ, UPDATE	676699d9-ac15-11eb-95c4-0242ac110002	Edit Delete

ALLOW ALL *

Unselect all

CREATE
 READ
 UPDATE
 DELETE

Success : Successfully created new group 25

Assigning an user to a group

← → ⌂ Not Secure | ec2-100-26-153-189.compute-1.amazonaws.com:8080/camunda/app/admin/default/#/authorization?resource=1

Camunda Admin Users Groups Tenants Authorizations System Demo Demo :

Dashboard » Authorizations

User Authorizations				
Type	User / Group	Permissions	Resource ID	Action
ALLOW	accounting	READ	demo	Edit Delete
ALLOW	accounting	READ	mary	Edit Delete
ALLOW	camunda-admin	ALL	*	Edit Delete
ALLOW	demo	ALL	demo	Edit Delete
ALLOW	john	ALL	john	Edit Delete
ALLOW	management	READ	demo	Edit Delete
ALLOW	management	READ	peter	Edit Delete
ALLOW	mary	ALL	mary	Edit Delete
ALLOW	peter	ALL	peter	Edit Delete
ALLOW	sales	READ	demo	Edit Delete
ALLOW	sales	READ	john	Edit Delete

[Create new authorization +](#)

User

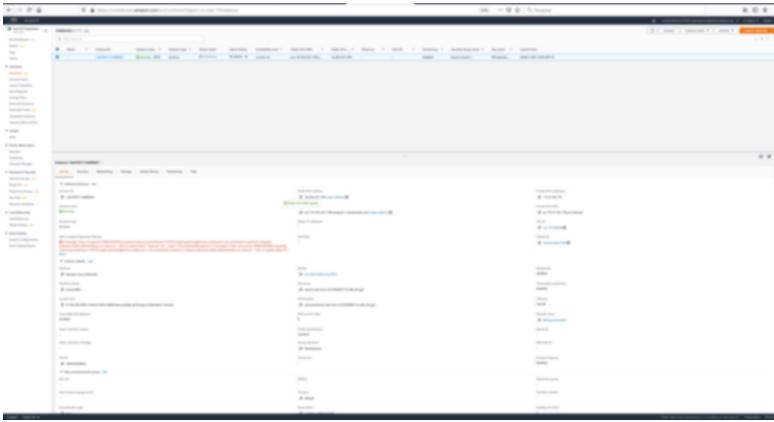
Success : ×
 Successfully created new group 25

Date and Time displayed in local timezone: Europe/Lisbon

Powered by Camunda Platform / v7.15.0

Camunda Engine Deployment

Camunda engine deployment on the cloud



```

sc2-user@ip-172-31-93-170:~$ 
Using username "sc2-user".
Authenticating with public key "SIPPreparation2021"
Last login Wed Nov 4 16:11:15 2020 from b112-152-37.dal.telepac.pt
[sc2-user@ip-172-31-93-170 ~]$ ls
Amazon Linux 2 AMI
[sc2-user@ip-172-31-93-170 ~]$ curl https://aws.amazon.com/amazon-linux-2/
curl: (28) Failed to connect to aws.amazon.com port 443: Connection timed out
[sc2-user@ip-172-31-93-170 ~]$ docker container ls --all
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
[sc2-user@ip-172-31-93-170 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[sc2-user@ip-172-31-93-170 ~]$ docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
1769a068557a        camunda/camunda-bpm-platform:latest   "/sbin/tini -- ./cam..."   46 hours ago       Exited (143) 46 hours ago          camunda
[sc2-user@ip-172-31-93-170 ~]$ docker container start 1769a068557a
[sc2-user@ip-172-31-93-170 ~]$ 

```

C1. Create an EC2 new instance of the type: Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type. The exact versions may change with time. And define the inbound rules to allow the 8080 and 22 accessed from anywhere.

C2. Install the most recent version of camunda-modeler using the download public available at:

- <https://camunda.com/download/modeler/>

C3. Access the new EC2 instance and execute the following commands:

```

sudo yum update -y
sudo yum install -y docker
sudo service docker start
sudo usermod -aG docker ec2-user
docker ps
exit

```

C4. Then, access again your EC2 instance and install and execute the Camunda engine with the following command:

```

docker pull camunda/camunda-bpm-platform:latest
docker run -d --name camunda -p 8080:8080 camunda/camunda-bpm-platform:latest

```

Test the installed Camunda engine opening the following URL in your browser (the default user/password is demo/demo):

```
# open browser with url: http://<YOUR\_AWS\_EC2\_NAME>:8080/camunda
```

Hint 1: Later if you would like to list all the available container use the following command:

```

PS C:\Users\Sérgio Guerreiro> docker container ls --all
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
44509250aa0c        camunda/camunda-bpm-platform:latest   "/sbin/tini -- ./cam..."   11 days ago       Exited
(143) 1 second ago

```

Hint 2: Later if you would like to stop the container use the following command:

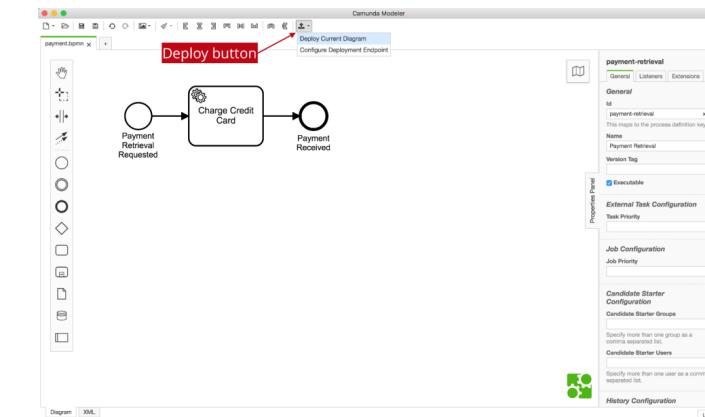
```
docker container stop <containerID_hash>
```

Hint 3: Later if you would like to restart the container use the following command:

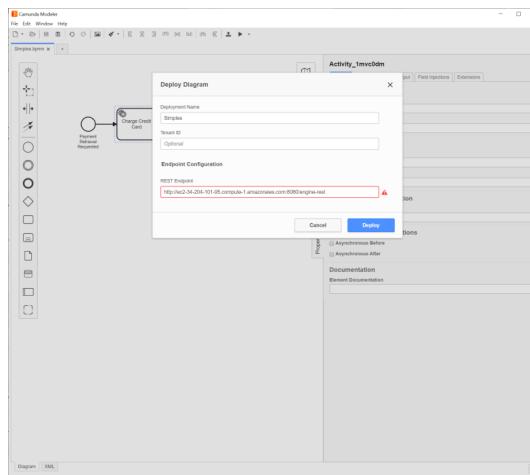
```
docker container start <containerID_hash>
```

D.1. Use the Camunda Modeler to Deploy the Process

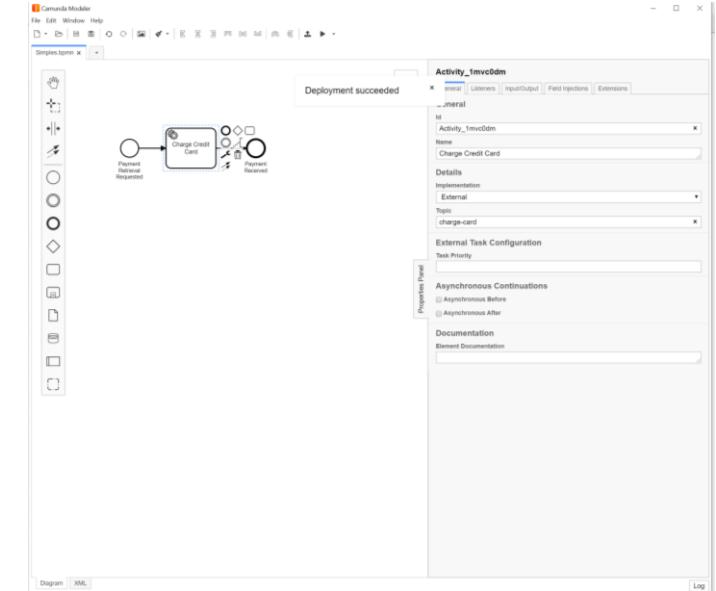
To deploy the Process, click on the deploy button in the Camunda Modeler, then give it the Deployment Name "Payment Retrieval" and click the Deploy button. From version 3.0.0 on, you will be required to provide an URL for an Endpoint Configuration along with Deployment Details. This can be either the root endpoint to the REST API (e.g. <http://localhost:8080/engine-rest>) or an exact endpoint to the deployment creation method (e.g. <http://localhost:8080/engine-rest/deployment/create>).



Deployment on the cloud



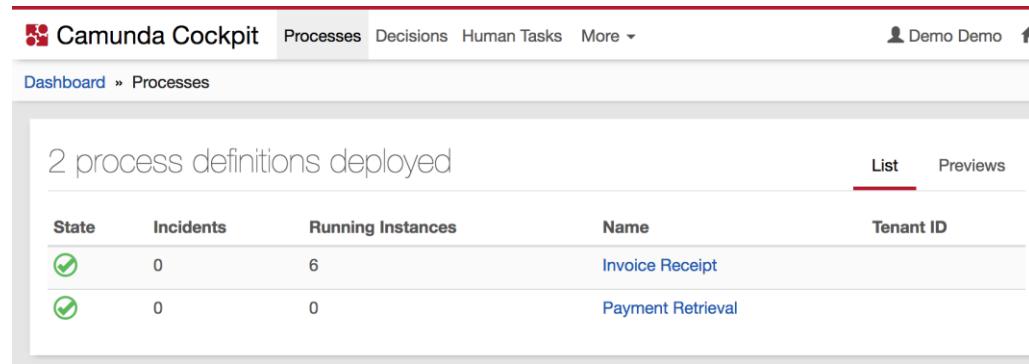
You should see a success message in the Camunda Modeler:



Deployment on the cloud

D.1. Verify the Deployment with Cockpit

Next, use Cockpit to see if the process was successfully deployed. Go to <http://localhost:8080/camunda/app/cockpit> and log in with the credentials demo / demo. Your process *Payment Retrieval* should be visible on the dashboard.



The screenshot shows the Camunda Cockpit interface. At the top, there is a navigation bar with the Camunda Cockpit logo, user information (Demo/Demo), and a home icon. Below the navigation bar, the URL `Dashboard > Processes` is visible. The main content area displays a table titled "2 process definitions deployed". The table has columns: State, Incidents, Running Instances, Name, and Tenant ID. There are two rows in the table:

State	Incidents	Running Instances	Name	Tenant ID
✓	0	6	Invoice Receipt	
✓	0	0	Payment Retrieval	

Deployment on the cloud

Screenshot of the Camunda Cockpit interface showing a process definition and its runtime details.

Definition Version: 1
Version Tag: null
Definition ID: Process_0z3xd3q1cab66215-2043-11eb-ac05-0242ac110002
Definition Key: Process_0z3xd3q
Definition Name: Payment Retrieval
History Time To Live: null
Tenant ID: null
Deployment ID: caacc523-2043-11eb-ac05-0242ac1...
Instances Running:

- current version: 0
- all versions: 0

The process diagram shows a sequence of events:
1. Start event → Payment Retrieval Requested
2. Task node: Charge Credit Card
3. End event: Payment Received

Process Instances tab selected. No process instances matched by current filter.

Date and Time displayed in local timezone: Europe/Lisbon

Powered by [camunda BPM](#) / v7.14.0

Camunda Cockpit

Camunda Cockpit

Right Now



8
Running Process Instances



0
Open Incidents



6
Open Human Tasks

Deployed

Process Definitions 2	Decision Definitions 2	Case Definitions 0	Deployments 2
--	---	---------------------------------------	----------------------------------

Date and Time displayed in local timezone: Europe/Lisbon

Powered by camunda BPM / v7.14.0

Camunda Cockpit

Camunda Cockpit - Processes Decisions Human Tasks More ▾

Dashboard > Processes > Invoice Receipt : Runtime

Definition Version: 2

Version Tag: V2.0

Definition ID: invoice:2:56e70454-1ebb-11eb-b03...

Definition Key: invoice

Definition Name: Invoice Receipt

History Time To Live: 45 days

Tenant ID: null

Deployment ID: 56da311-1ebb-11eb-b032-0242ac...

Instances Running:

- current version: 3
- all versions: 6

```

graph TD
    TA((Team Assistant)) --> AA[Assign Approver Group]
    AP((Approver)) --> AI[Approve Invoice]
    AC((Accountant)) --> PB[Prepare Bank Transfer]
    
    AA --> RI[Review Invoice]
    RI --> DS1{Review successful?}
    DS1 -- yes --> AI
    DS1 -- no --> IP1((Invoice not processed))
    
    AI --> DS2{Invoice approved?}
    DS2 -- yes --> PB
    DS2 -- no --> IP2((Invoice not processed))
    
    PB --> AI2[Archive Invoice]
    AI2 --> IP3((Invoice processed))
    
    subgraph FAS [Financial Accounting System]
        direction TB
        FAS1[ ] --- FAS2[ ]
        FAS2 --- FAS3[ ]
    end
    
    FAS3 --- RI
    FAS3 --- DS1
    FAS3 --- DS2
    FAS3 --- AI2
  
```

Activity Instance Statistics: on

Process Instances Incidents Called Process Definitions Job Definitions

Add criteria			
State	ID	Start Time	Business Key
✓	59242232-1ebb-11eb-b032-0242ac110002	2020-11-04T16:32:36	
✓	598fdcd3-1ebb-11eb-b032-0242ac110002	2020-10-30T16:32:37	
✓	593a1b68-1ebb-11eb-b032-0242ac110002	2020-10-21T17:32:36	

Powered by camunda BPM / v7.14.0

Instantiating one process model

Instantiating one process model

D.1. Start a process invocation of your BPMN using curl:

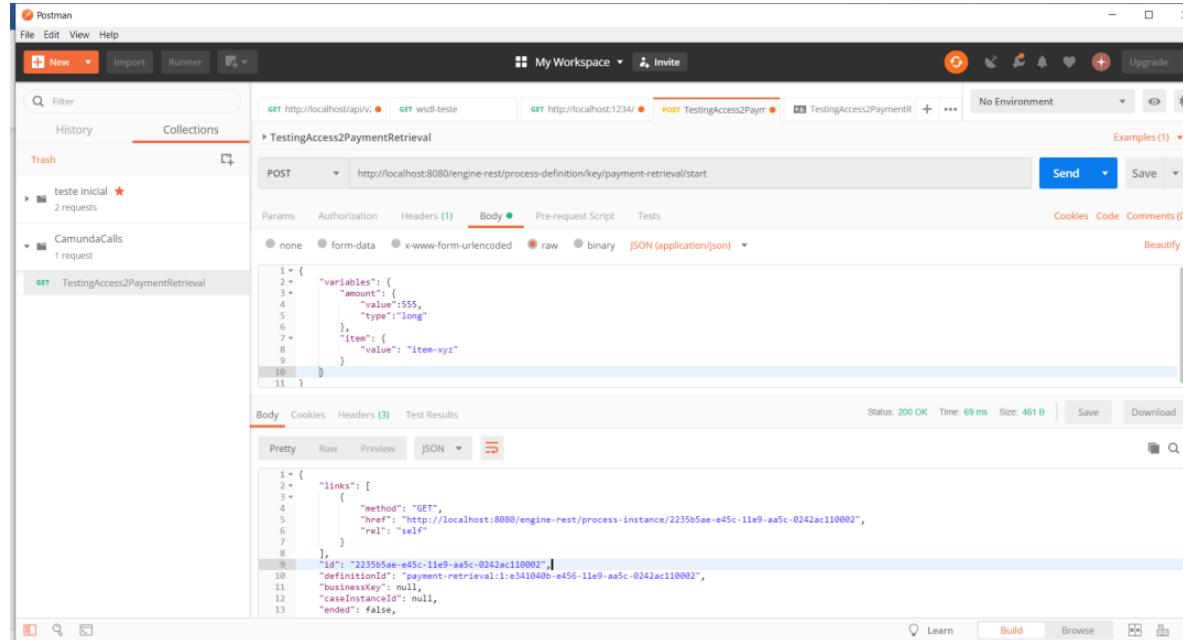
```
curl -H "Content-Type: application/json" -X POST -d '{"variables": {"amount": {"value":555,"type":"long"}, "item": {"value":"item-xyz"} } }' http://localhost:8080/engine-rest/process-definition/key/payment-retrieval/start
```

Running example

```
Sérgio Guerreiro@LAPTOP-DF942S8T ~
$ curl -H "Content-Type: application/json" -X POST -d '{"variables": {"amount": {"value":555,"type":"long"}, "item": {"value":"item-xyz"} } }' http://localhost:8080/engine-rest/process-definition/key/payment-retrieval/start
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
          Dload  Upload   Total Spent  Left Speed
100  431    0  345  100    86  345      86  0:00:01 --:--:--  0:00:01  5525
{"links": [{"method": "GET", "href": "http://localhost:8080/engine-rest/process-instance/a5c230ce-e459-11e9-aa5c-0242ac110002", "rel": "self"}], "id": "a5c230ce-e459-11e9-aa5c-0242ac110002", "definitionId": "payment-retrieval:1:e341040b-e456-11e9-aa5c-0242ac110002", "businessKey": null, "caseInstanceId": null, "ended": false, "suspended": false, "tenantId": null}
```

Instantiating one process model

Alternatively, install postman <https://www.getpostman.com/downloads/> (for windows OS and Mac OS). Here is what the request might look like in Postman:



The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'File', 'Edit', 'View', 'Runner', and a dropdown for 'My Workspace'. Below the workspace dropdown, there are icons for 'Invite', 'Upgrade', and 'No Environment'. The main workspace shows a list of collections: 'teste inicial' (with 2 requests) and 'CamundaCalls' (with 1 request). The selected collection is 'TestingAccess2PaymentRetrieval'.

The request details are as follows:

- Method:** POST
- URL:** http://localhost:8080/engine-rest/process-definition/key/payment-retrieval/start
- Body:** JSON (application/json)
- Body Content:**

```
1 {  
2   "variables": {  
3     "amount": {  
4       "value": 555,  
5       "type": "long"  
6     },  
7     "item": {  
8       "value": "item-xyz"  
9     }  
10 }  
11 }
```

Below the body, the response is displayed in JSON format:

```
1 {  
2   "links": [  
3     {  
4       "method": "GET",  
5       "href": "http://localhost:8080/engine-rest/process-instance/2235b5ae-e45c-11e9-aa5c-0242ac110002",  
6       "rel": "self"  
7     }  
8   ],  
9   "id": "2235b5ae-e45c-11e9-aa5c-0242ac110002",  
10  "definitionId": "payment-retrieval:1:e341040b-e456-11e9-aa5c-0242ac110002",  
11  "businessKey": null,  
12  "businessKey": null,  
13  "caseInstanceId": null,  
14  "ended": false,
```

Note1: due to the asynchronous JAVA service, the instance could be listed as an active instance for a while in the camunda cockpit.

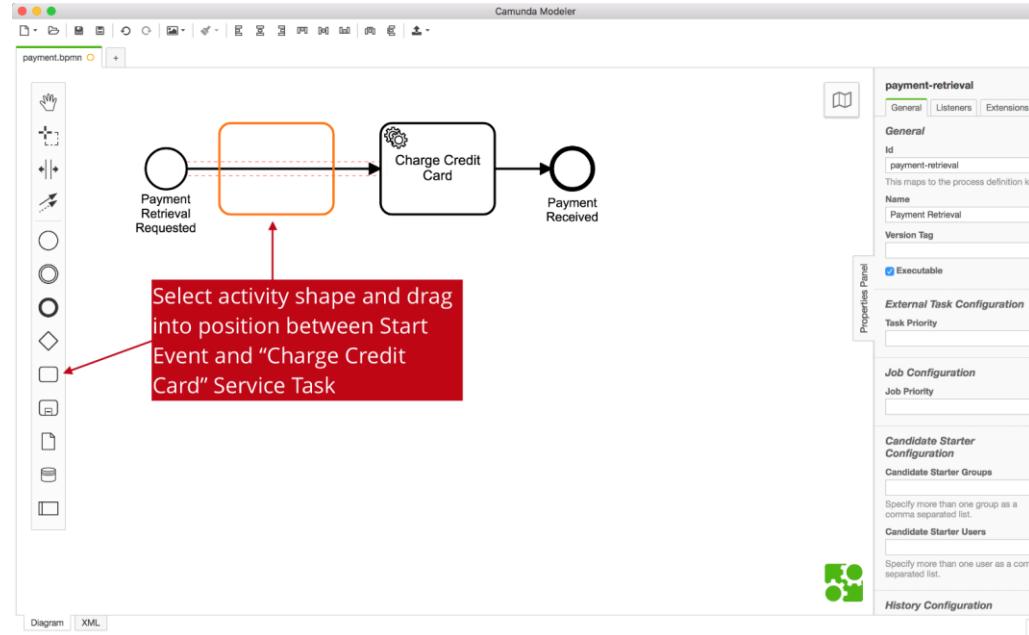
Note2: the external java client is not accessible remotely. It uses the org.camunda.bpm.client.ExternalTaskClient class which takes care of the invocation directly.

Camunda Integrated Forms

A user interaction task

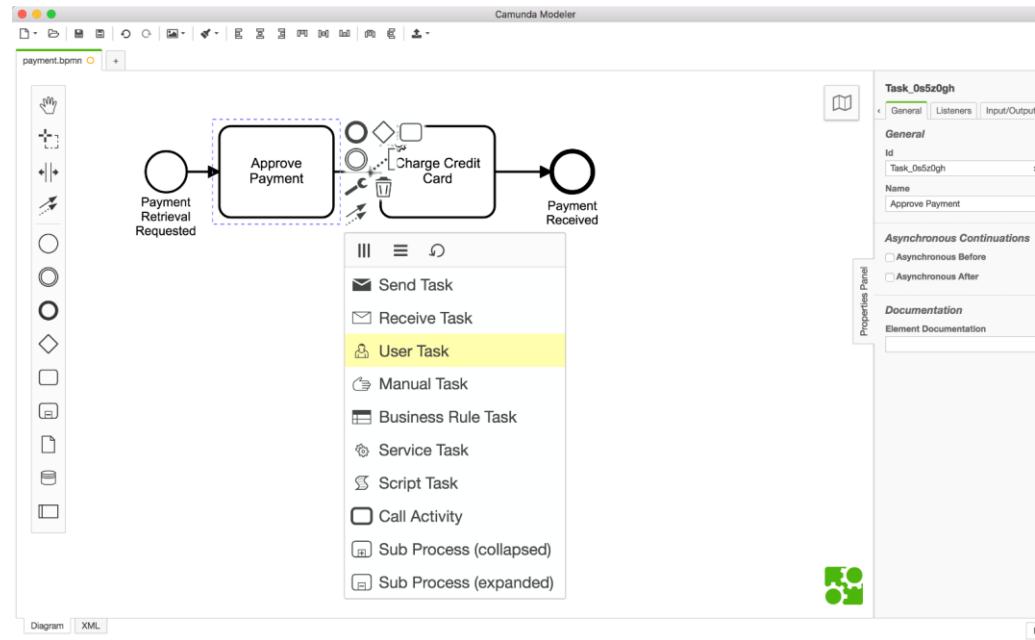
E1. Add a User Task

We want to modify our process so that we can involve humans. To do so, open the process in the Camunda Modeler. Next, from the Modeler's left-hand menu, select the activity shape (rectangle) and drag it into position between the Start Event and the "Charge Credit Card" Service Task. Name it *Approve Payment*.



A user interaction task

Change the activity type to *User Task* by clicking on it and using the wrench button menu.

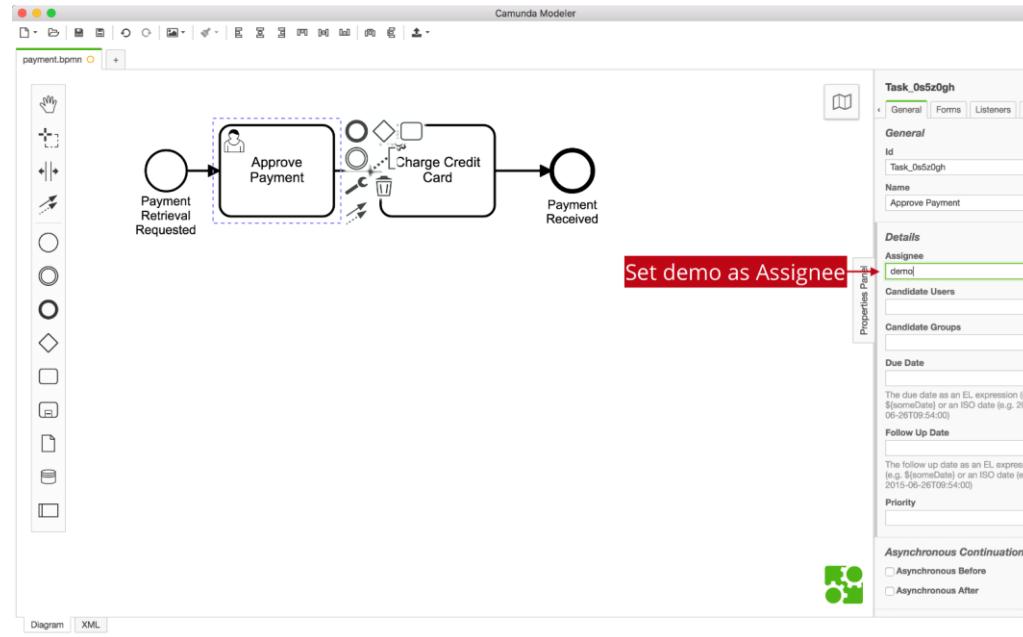


A user interaction task

E1. Configure a User Task

Next, open the properties view. If the properties view is not already visible, click on the “Properties Panel” label on the right-hand side of the Modeler canvas.

Select the User Task on the canvas. This will update the selection in the properties view. Scroll to the property named **Assignee**. Type *demo*.



A user interaction task

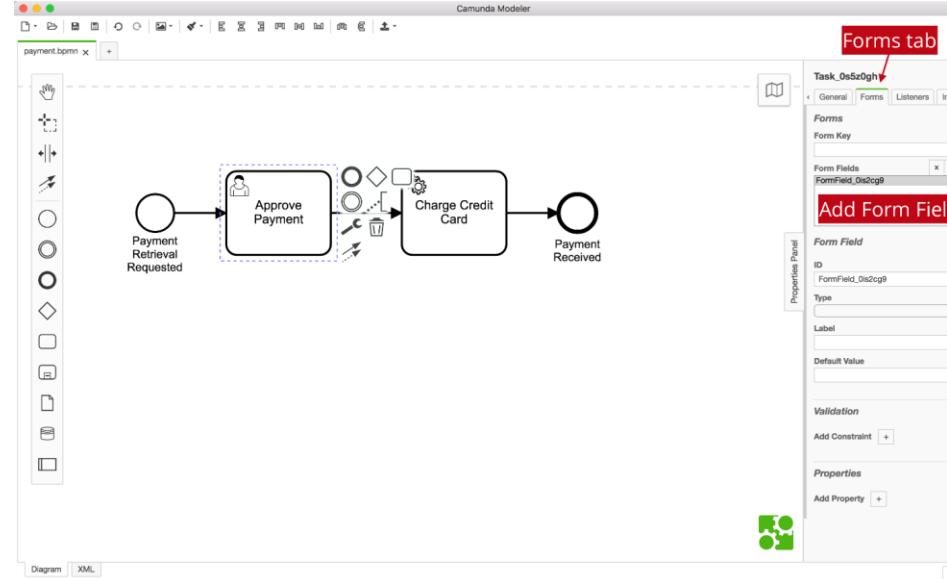
E.1. Configure a basic form in the User Task

This step will also take place in the properties panel. If the panel is not already visible, click on the “Properties Panel” label on the right-hand side of the Modeler canvas.

Select the User Task on the canvas. This will update the selection in the properties view.

Click on the Tab **Forms** in the properties panel.

Add three form fields by clicking on the plus button:



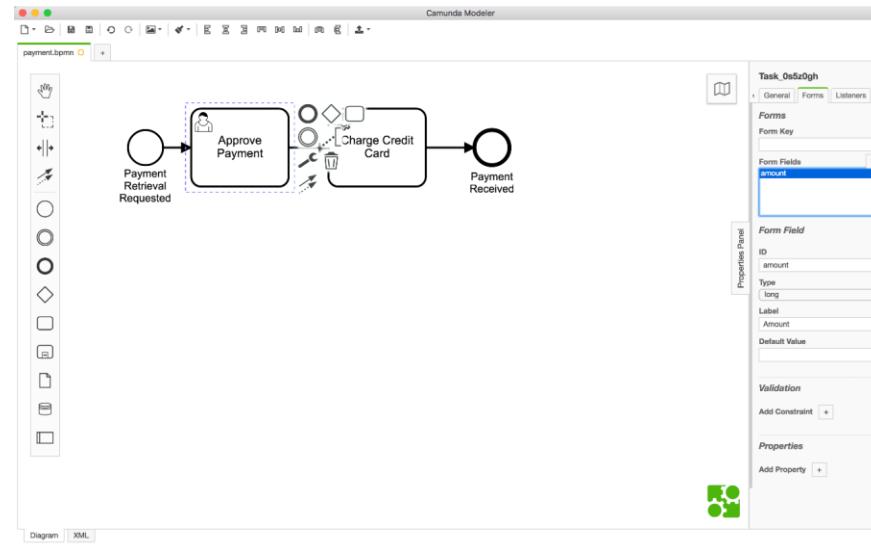
A user interaction task

Field 1:

ID: amount

Type: long

Label: Amount



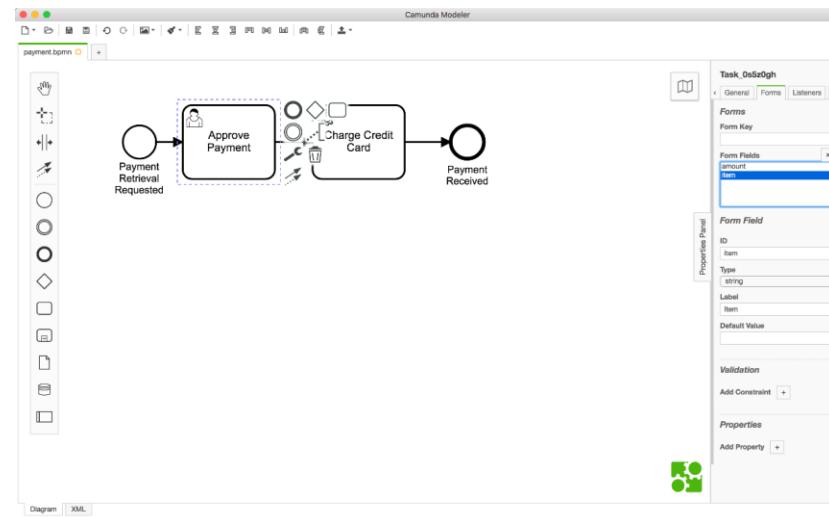
A user interaction task

Field 2:

ID: item

Type: string

Label: Item



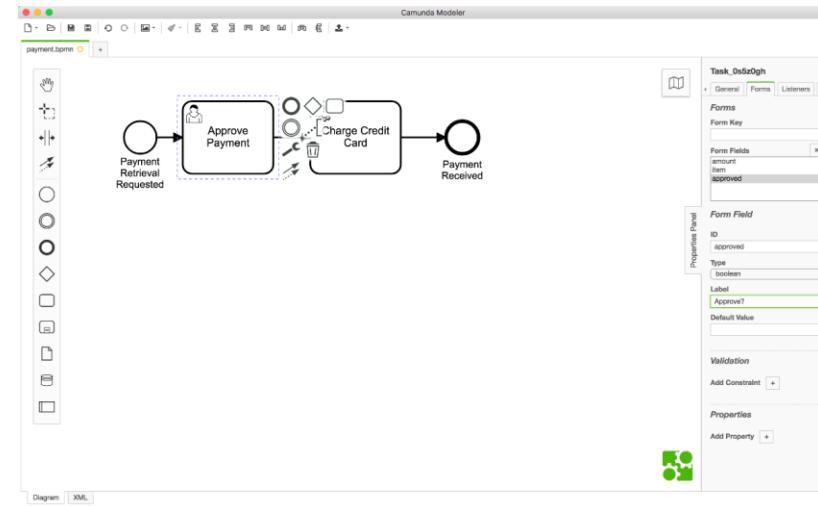
A user interaction task

Field 3:

ID: approved

Type: boolean

Label: Approved?



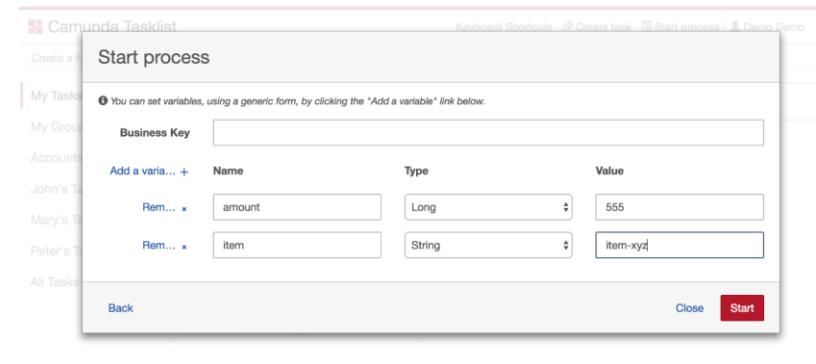
A user interaction task

E.1. Deploy the Process

Use the [Deploy](#) Button in the Camunda Modeler to deploy the updated process to Camunda.

E.2. Work on the Task

Go to Tasklist (<http://localhost:8080/camunda/app/tasklist>) and log in with the credentials “demo / demo”. Click on the button to start a process instance. This opens a dialog where you can select *Payment Retrieval* from the list. Now you can set variables for the process instance using a generic form.



The screenshot shows the Camunda Tasklist interface with a modal dialog titled "Start process". The dialog contains instructions: "You can set variables, using a generic form, by clicking the "Add a variable" link below." Below this, there is a table for defining variables:

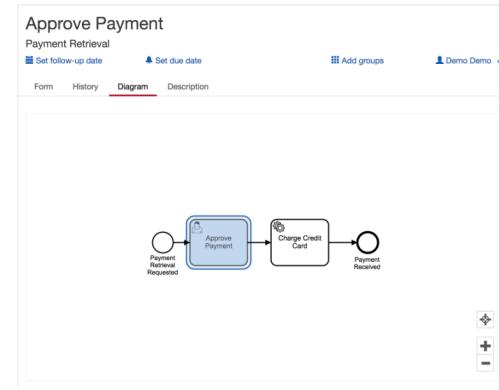
Add a varia... +	Name	Type	Value
Rem... *	amount	Long	555
Rem... *	item	String	item-xyz

At the bottom of the dialog are "Back", "Close", and "Start" buttons.

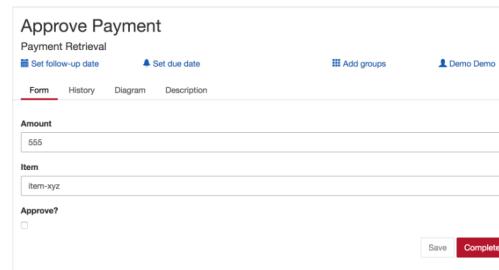
The generic form can be used whenever you have not added a dedicated form for a User Task or a Start Event. Click on the *Add a variable* button to create a new row. Fill in the form as shown in the screenshot. When you're done, click *Start*.

A user interaction task

You should now see the *Approve Payment* task in your Tasklist. Select the task and click on the *Diagram* tab. This displays the process diagram highlighting the User Task that's waiting to be worked on.



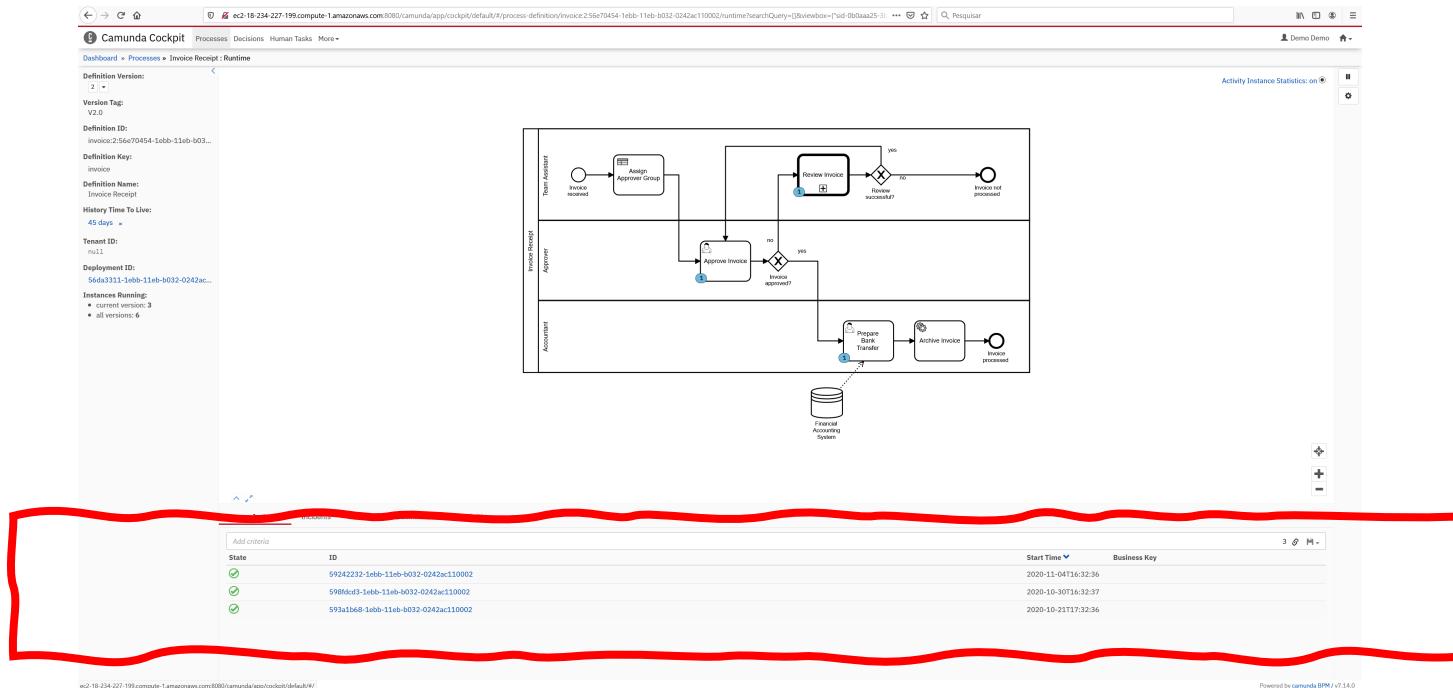
To work on the task, select the *Form* tab. Because we defined the variables in the Form Tab in the Camunda Modeler, the Tasklist has automatically generated form fields for us.



The form tab displays several input fields and buttons. At the top, there are tabs for "Form", "History", "Diagram", and "Description", with "Form" being the active tab. Below the tabs, there are two input fields: "Amount" containing the value "555" and "Item" containing the value "Item-xyz". Underneath these fields is a checkbox labeled "Approve?". At the bottom right of the form are two buttons: "Save" and a larger red button labeled "Complete".

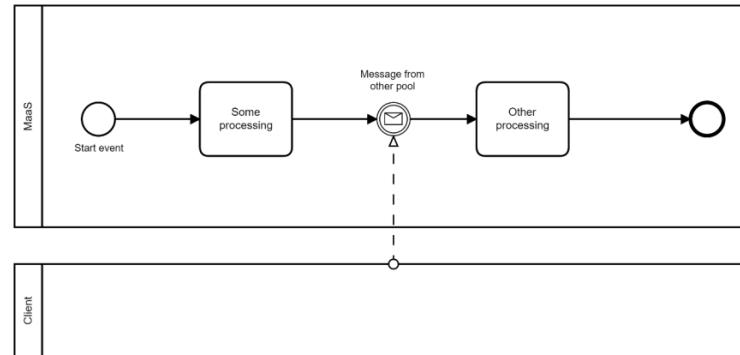
Multiple instances of a process

Multiple instances of the same process

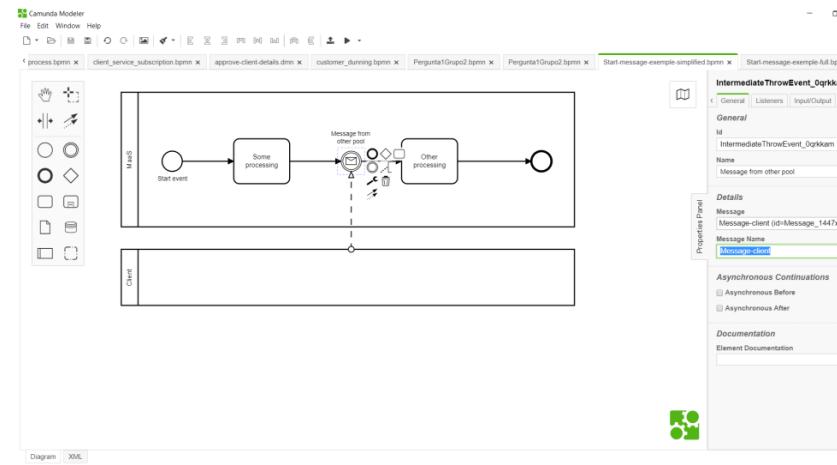


Multiple instances - 5.2. Correlation ID of a process to deal with messages

J1. Design the following collaboration in Camunda Modeler:



Where the intermediate message catch event should have a configuration similar with the following:



Multiple instances - 5.2. Correlation ID of a process to deal with messages

J1. Deploy the model in your Camunda engine.

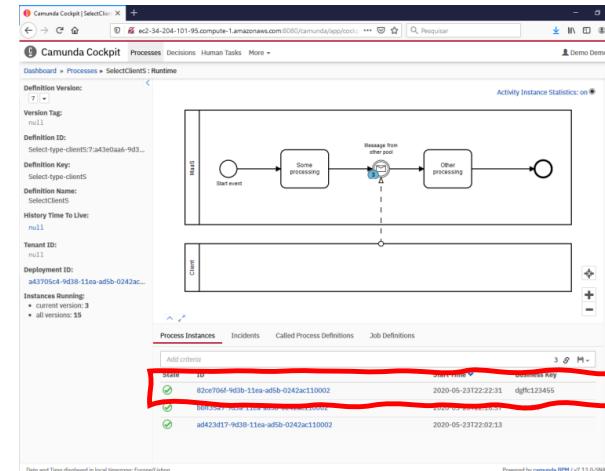
J2. Start the process with a command similar with the following:

```
curl -H "Content-Type: application/json" --data "@bodyStart.json" -X POST http://ec2-34-204-101-95.compute-1.amazonaws.com:8080/engine-rest/process-definition/key>Select-type-clientS/start
```

where bodyStart.json defines a key for that new instance of the collaboration:

```
{  
    "businessKey" : "dgfffc123455"  
}
```

You can verify the creation of the instance in the camunda cockpit. You will obtain something similar with the following business key as defined:



Multiple instances - 5.2. Correlation ID of a process to deal with messages

J1. Then, the instance is expecting a message. To test the catch of a REST message, use a similar curl command:

```
curl -H "Content-Type: application/json" --data "@body.json" -X POST http://ec2-34-204-101-95.compute-1.amazonaws.com:8080/engine-rest/message
```

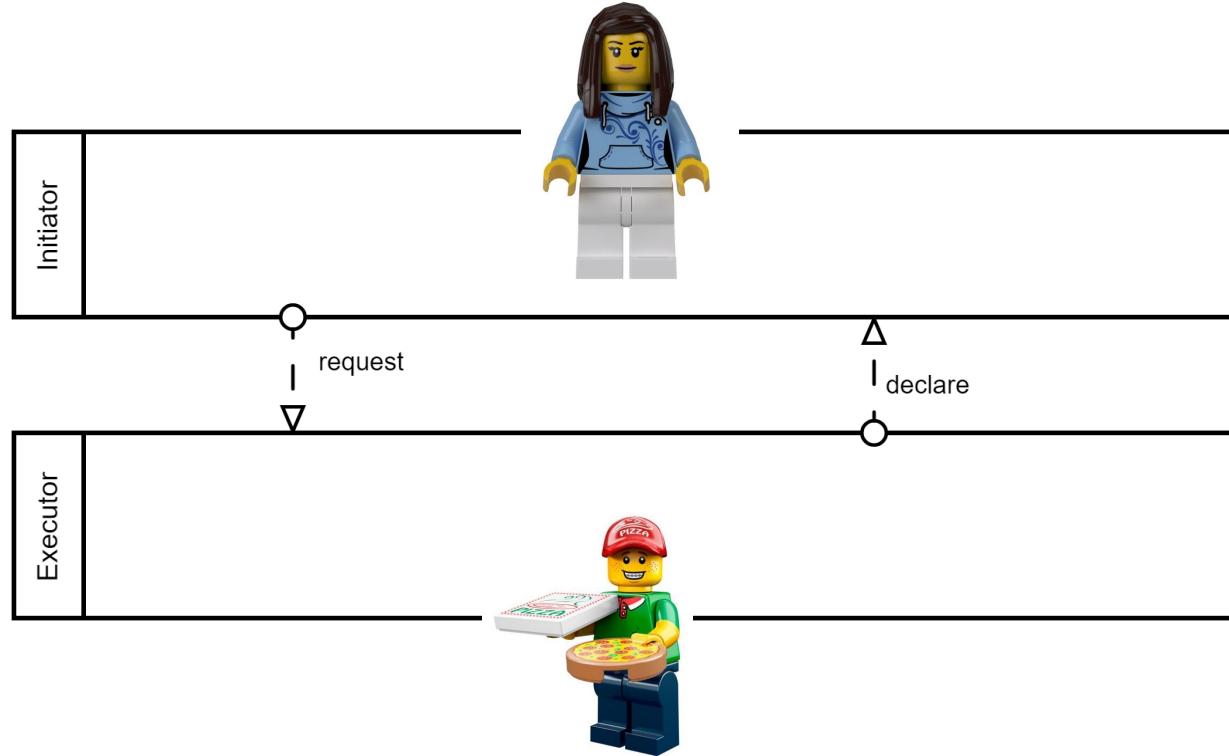
where bodyStart.json defines the message listener and the correlation with the key for your instance:

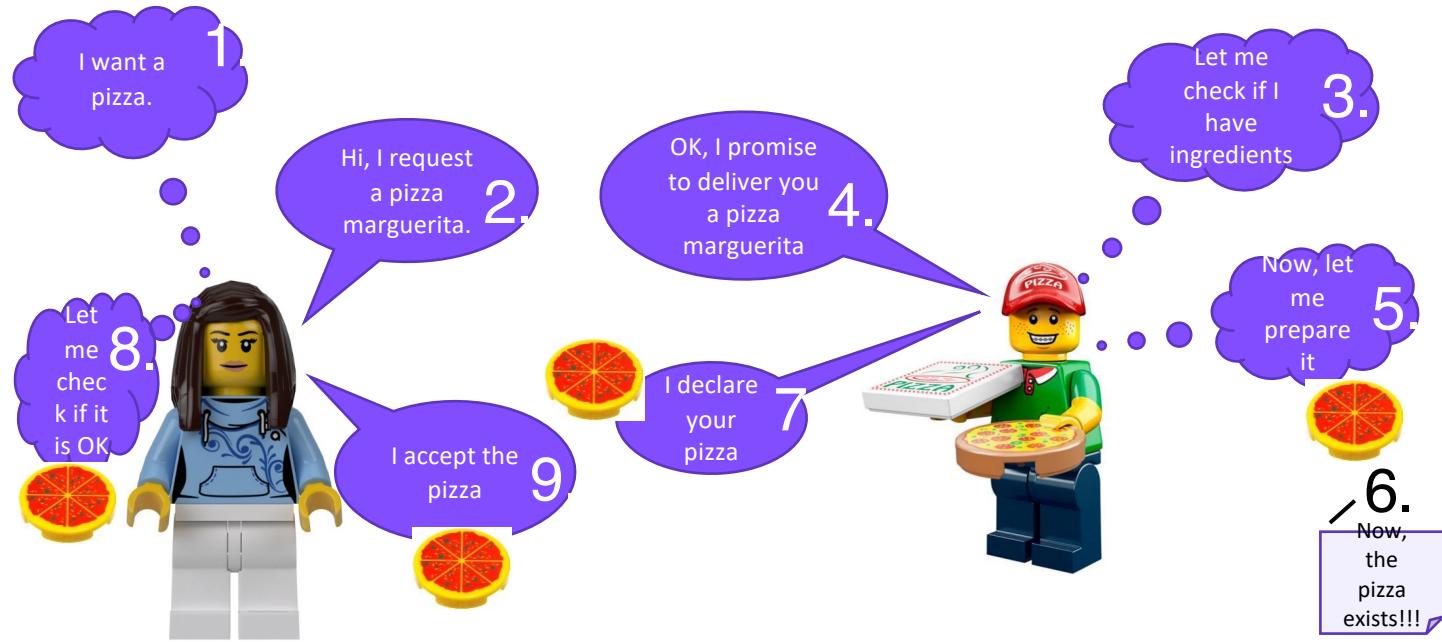
```
{  
  "messageName" : "Message-client",  
  "businessKey" : "dgfffc123455",  
  "resultEnabled" : true  
}
```

Process discovery and Enrichment

A BPMN pattern oriented solution

Mapping DEMO complete pattern onto BPMN



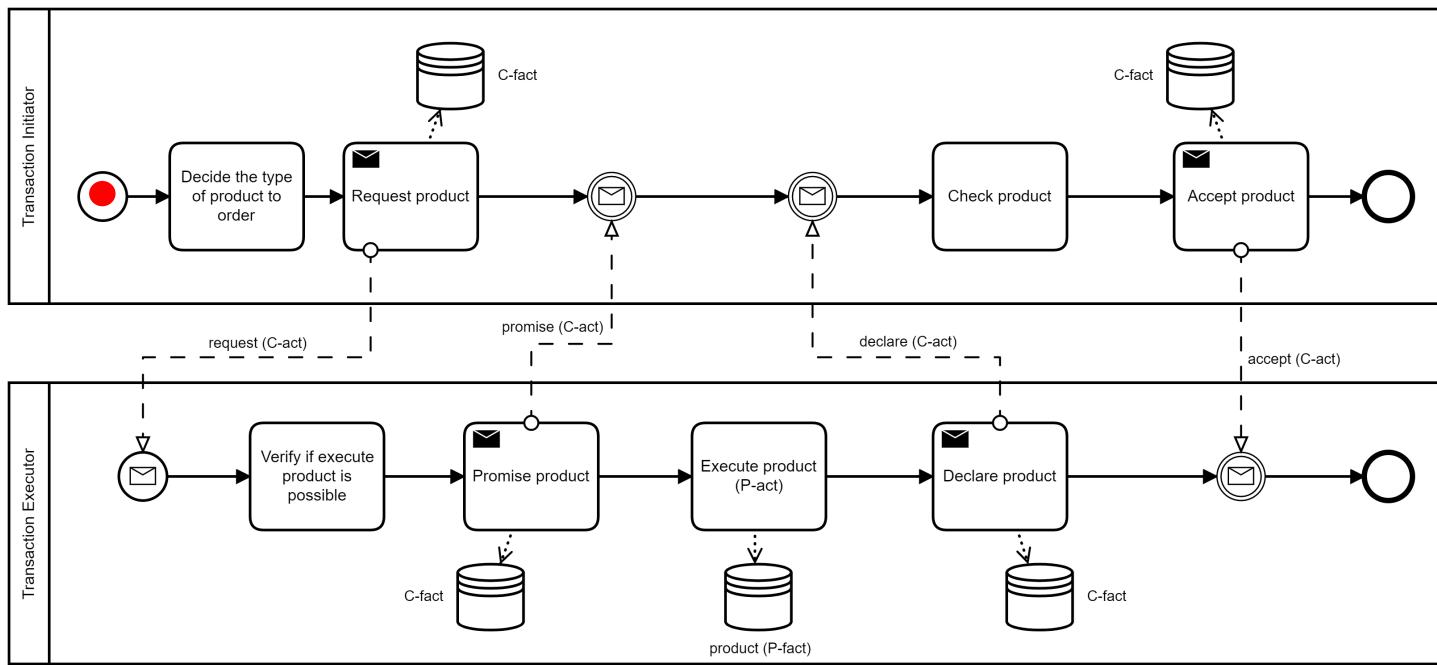


2 ACTORS COMMUNICATING...what happens...
if everything runs happily?



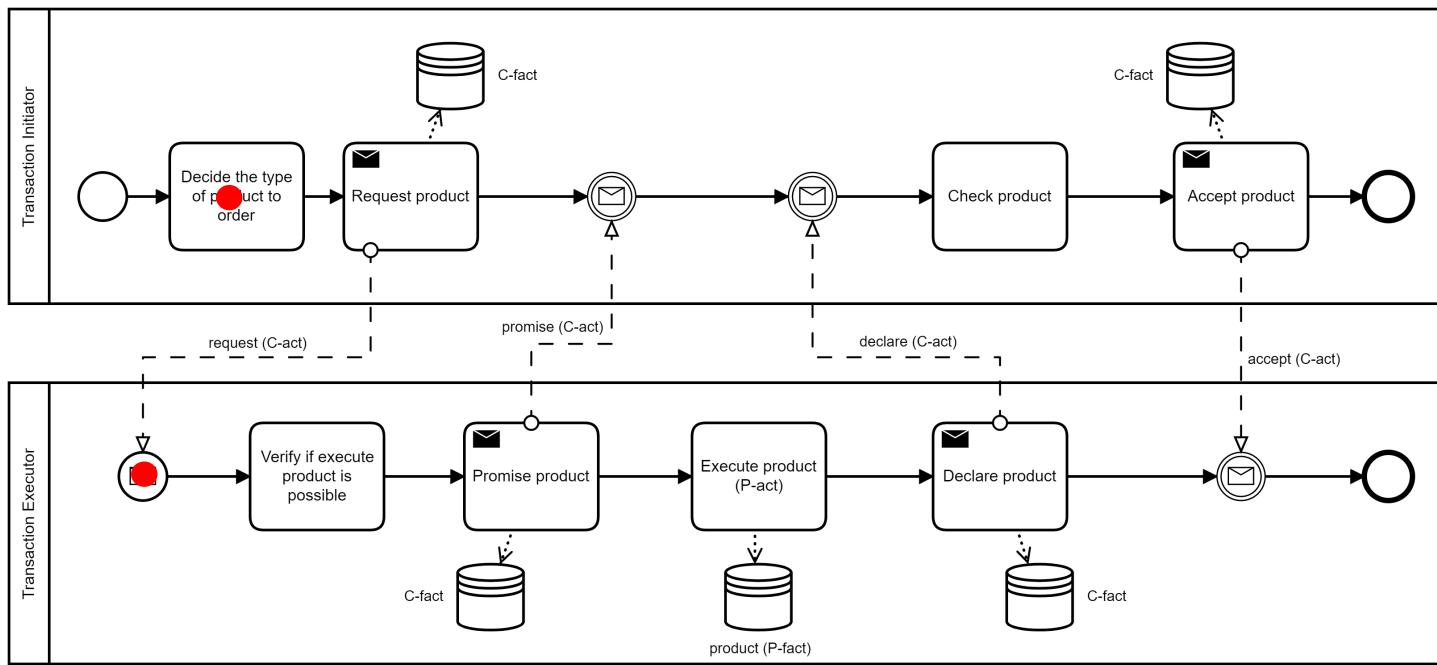
**Simulating the
Communication pattern
between 2 participants,
as a social system...**

In BPMN



I want a
pizza. 1

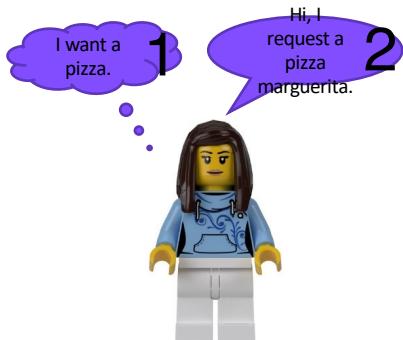
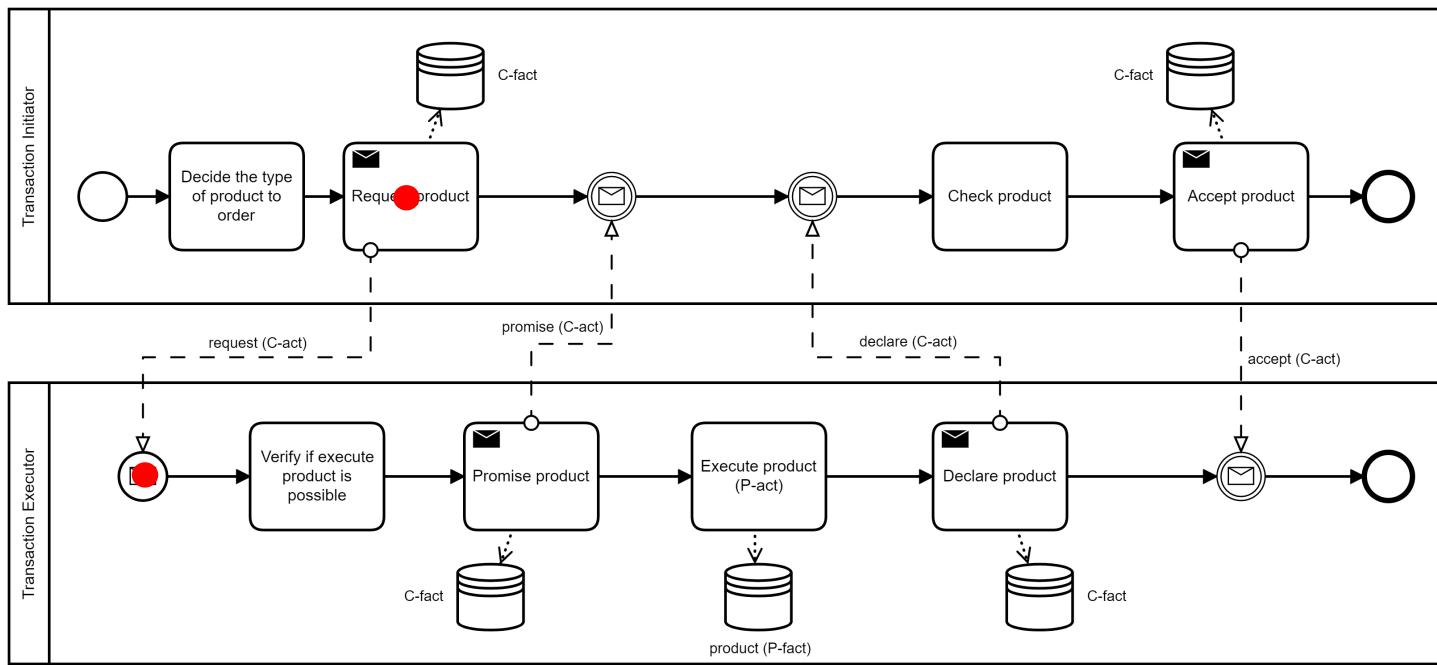


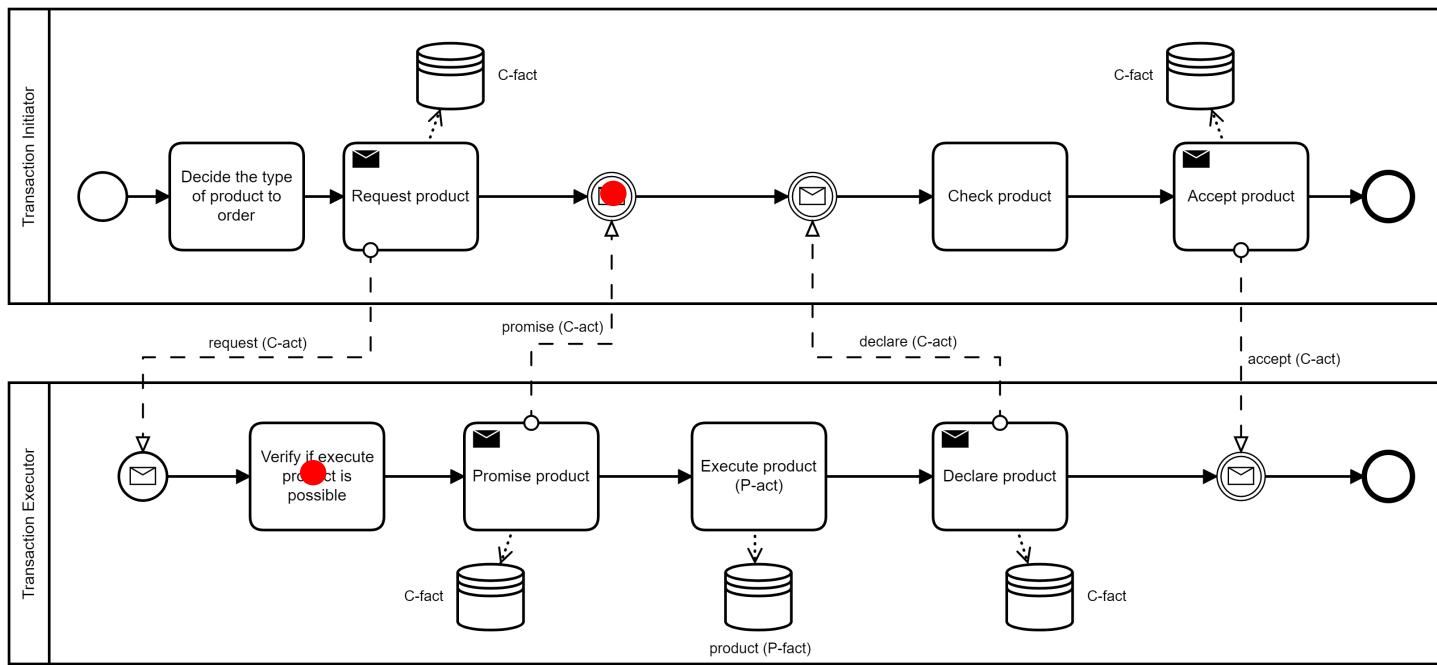


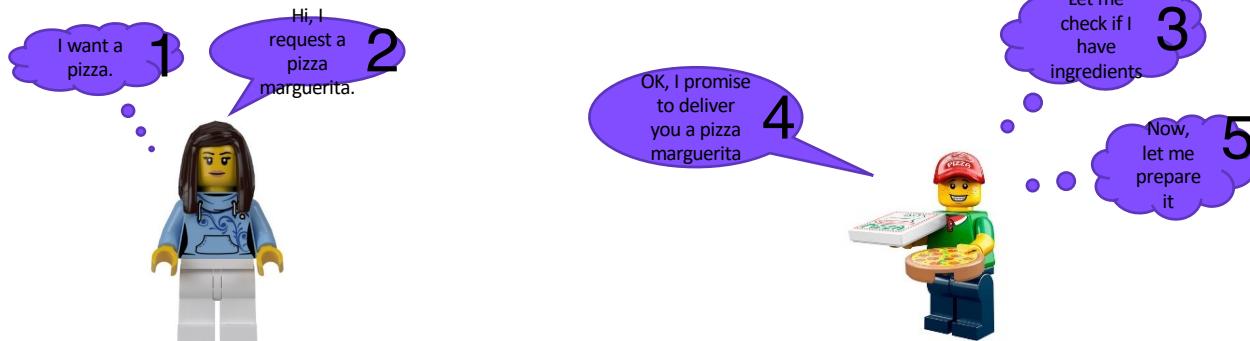
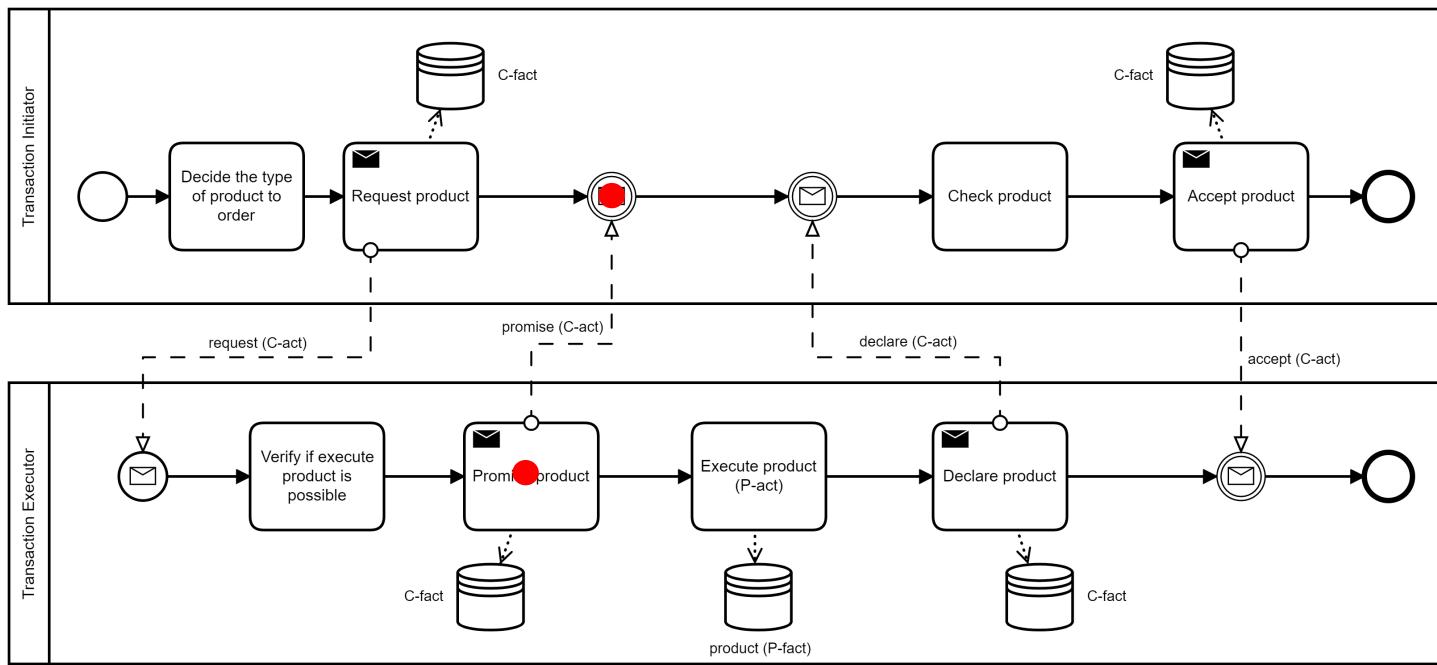
I want a pizza. **1**

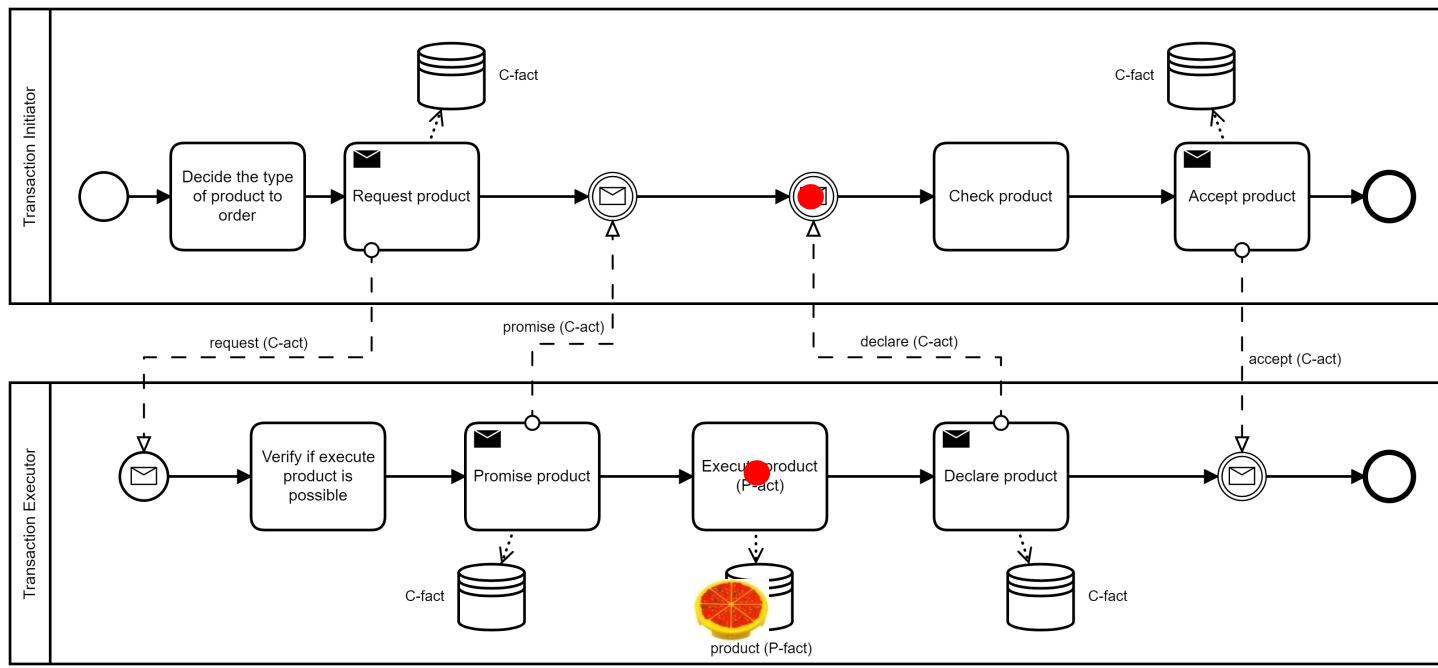
Hi, I request a pizza marguerita. **2**

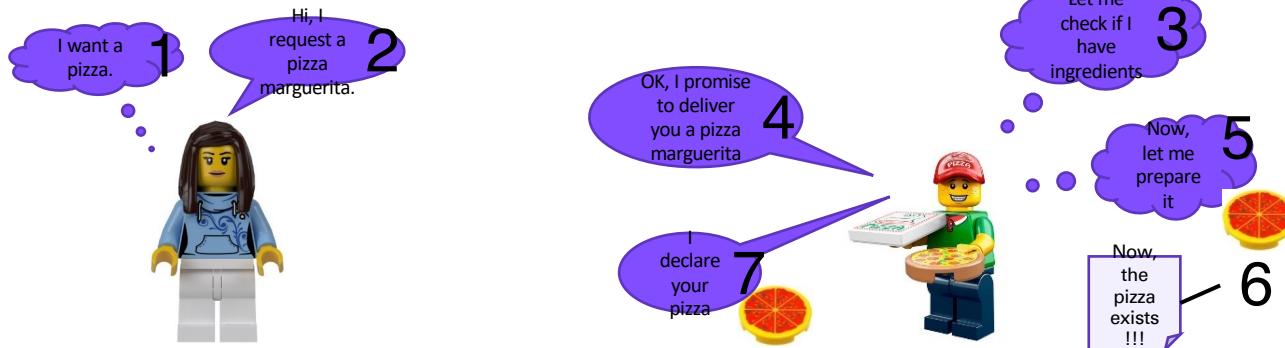
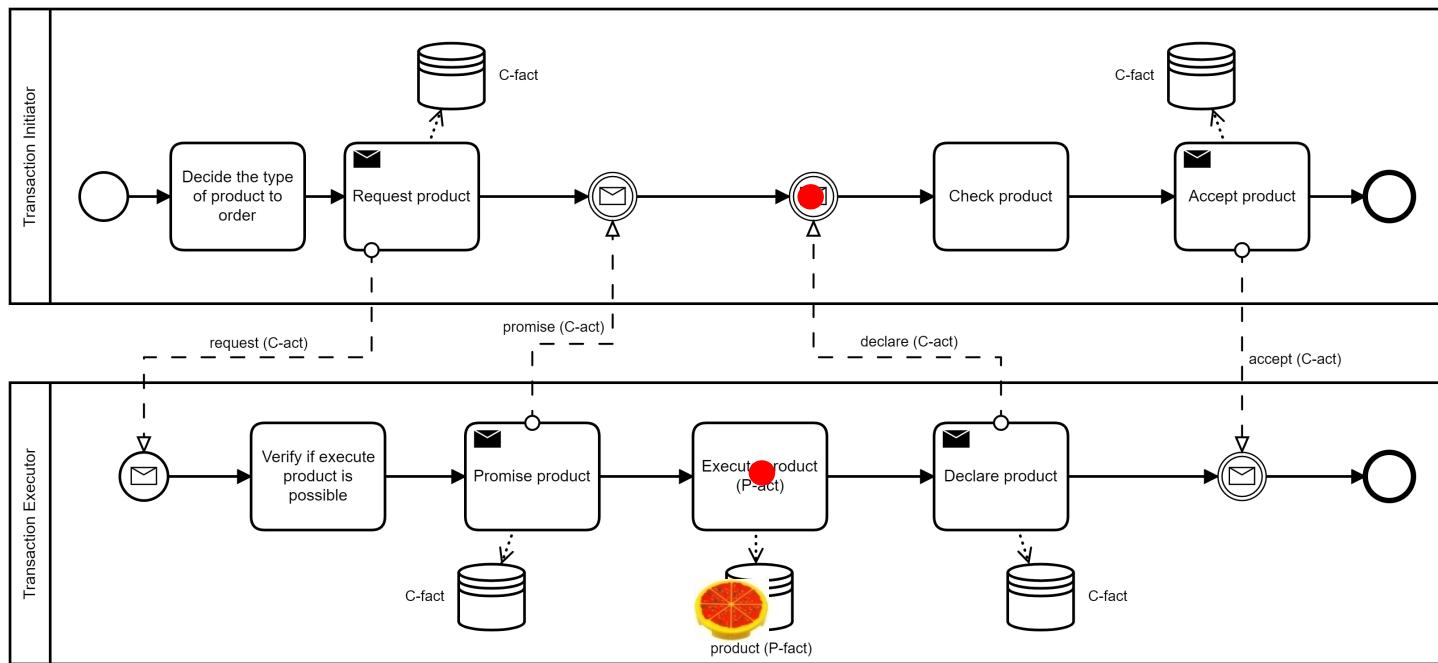


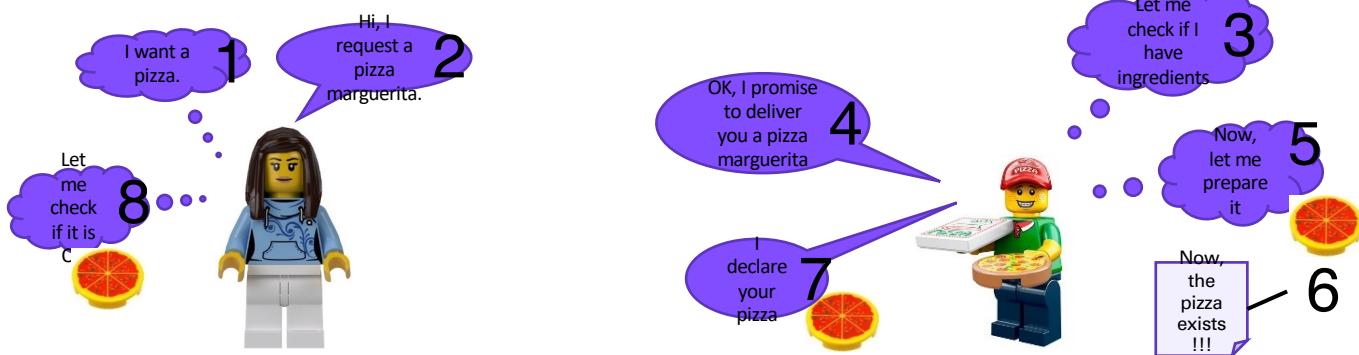
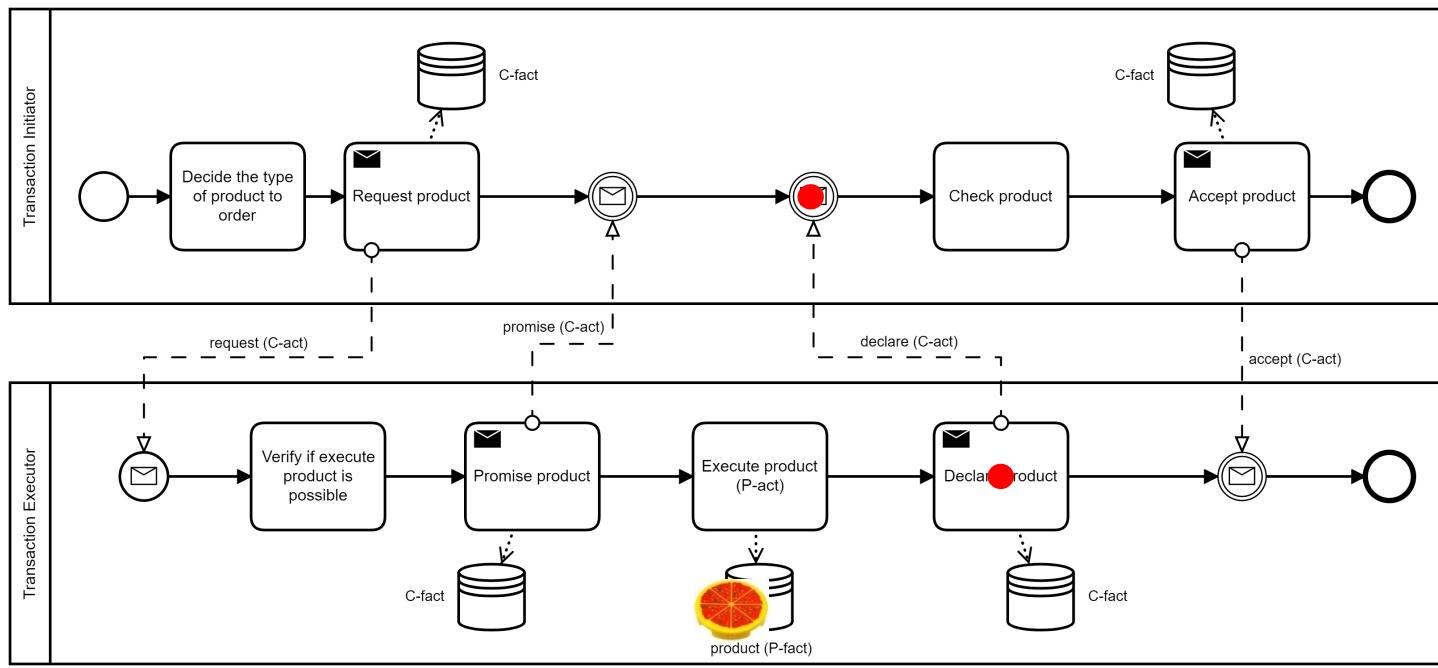


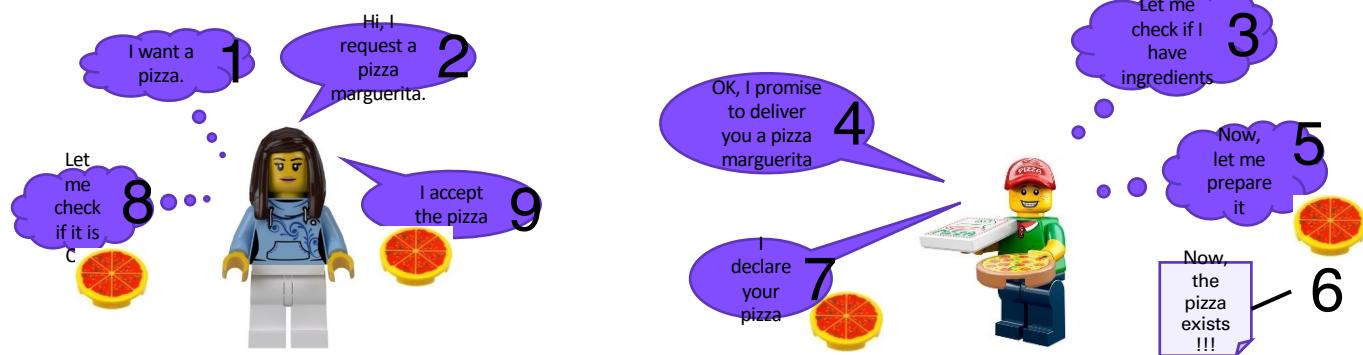
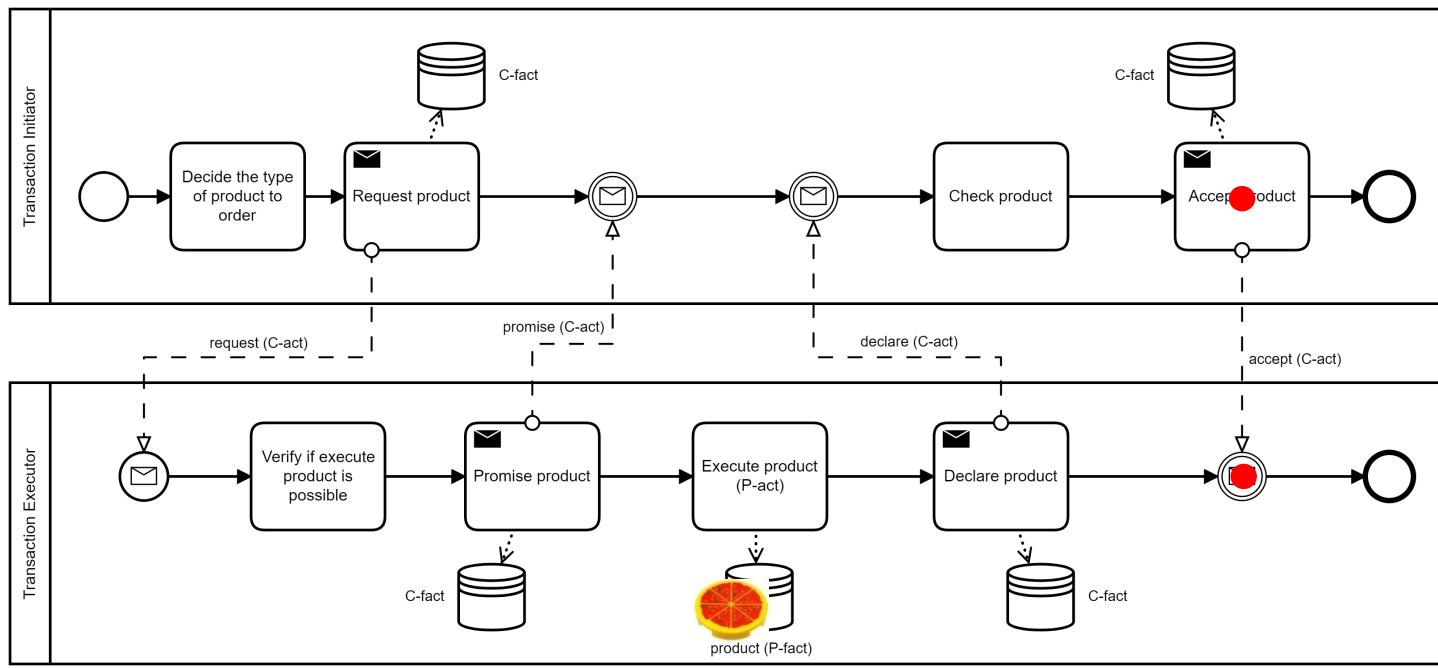






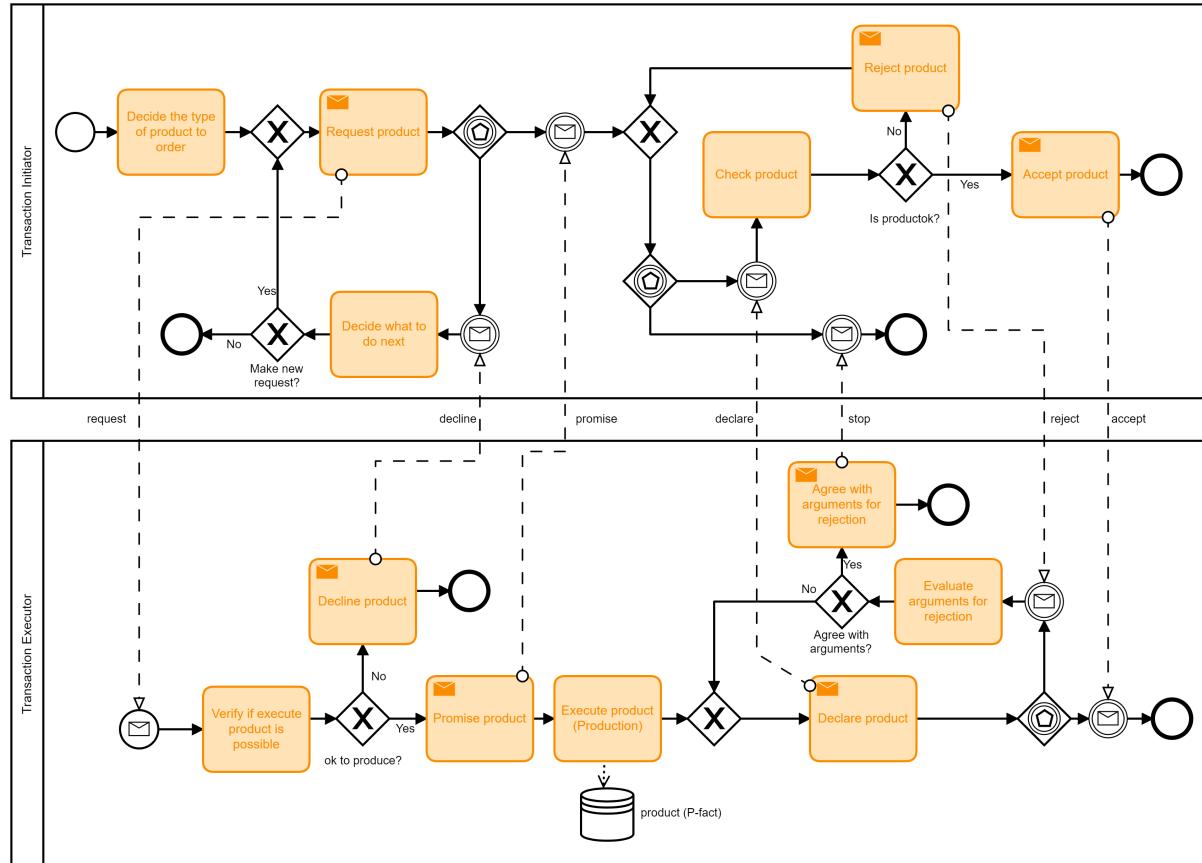








Enrichment – business transaction pattern – happy flow + rejections + declinations



13 variability points =>

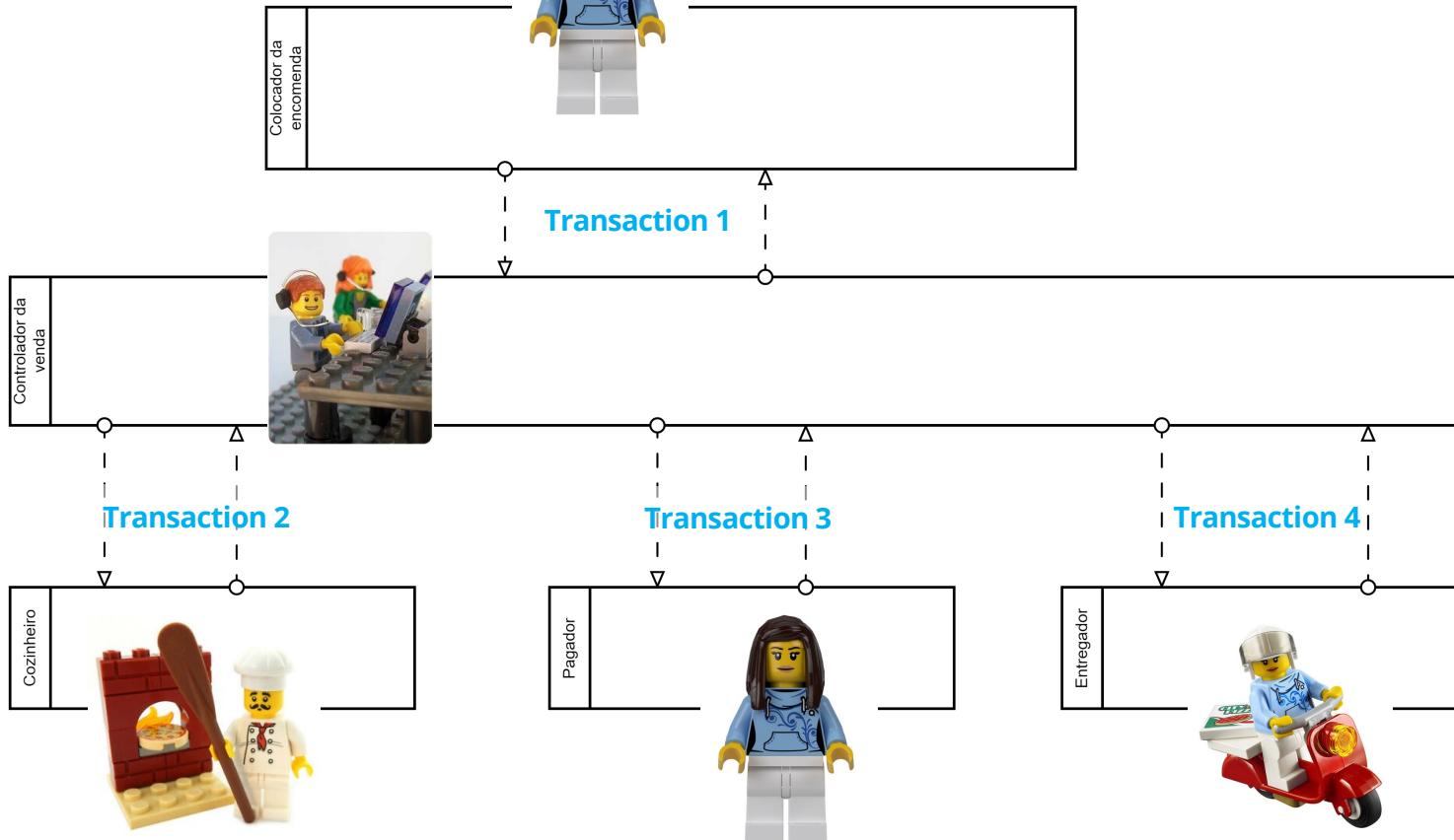
$$2^{13} =$$

8192

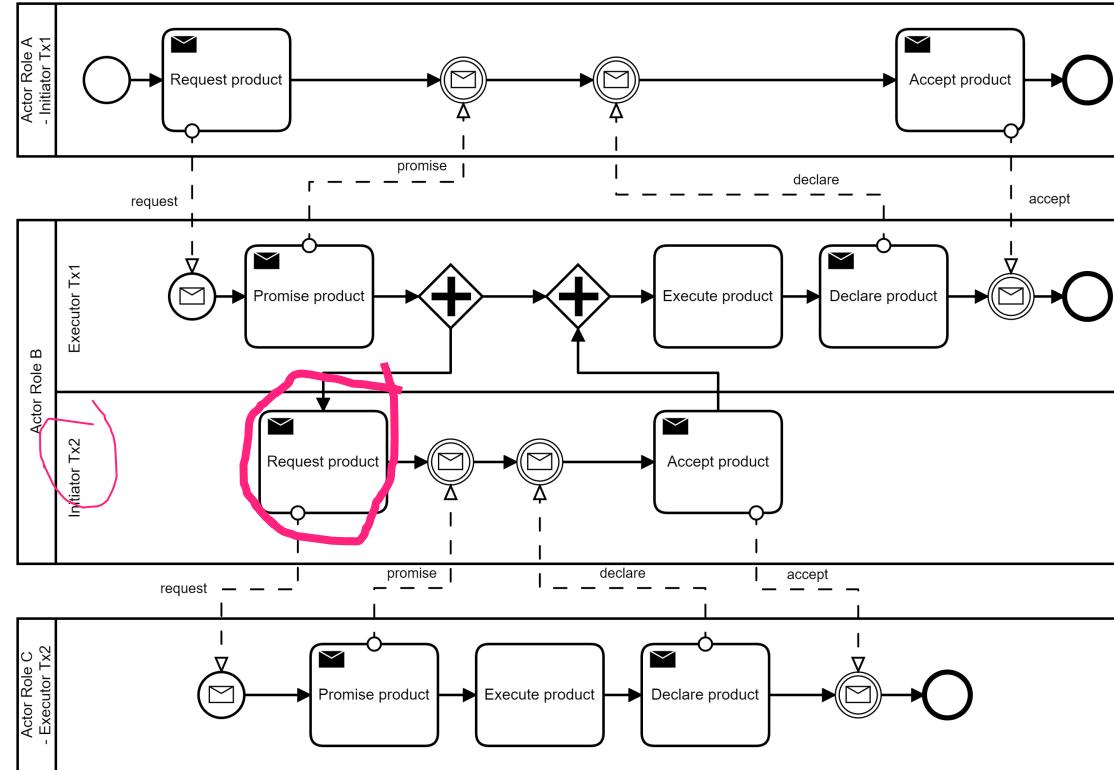
**process
configurations**



Composition of transactions...

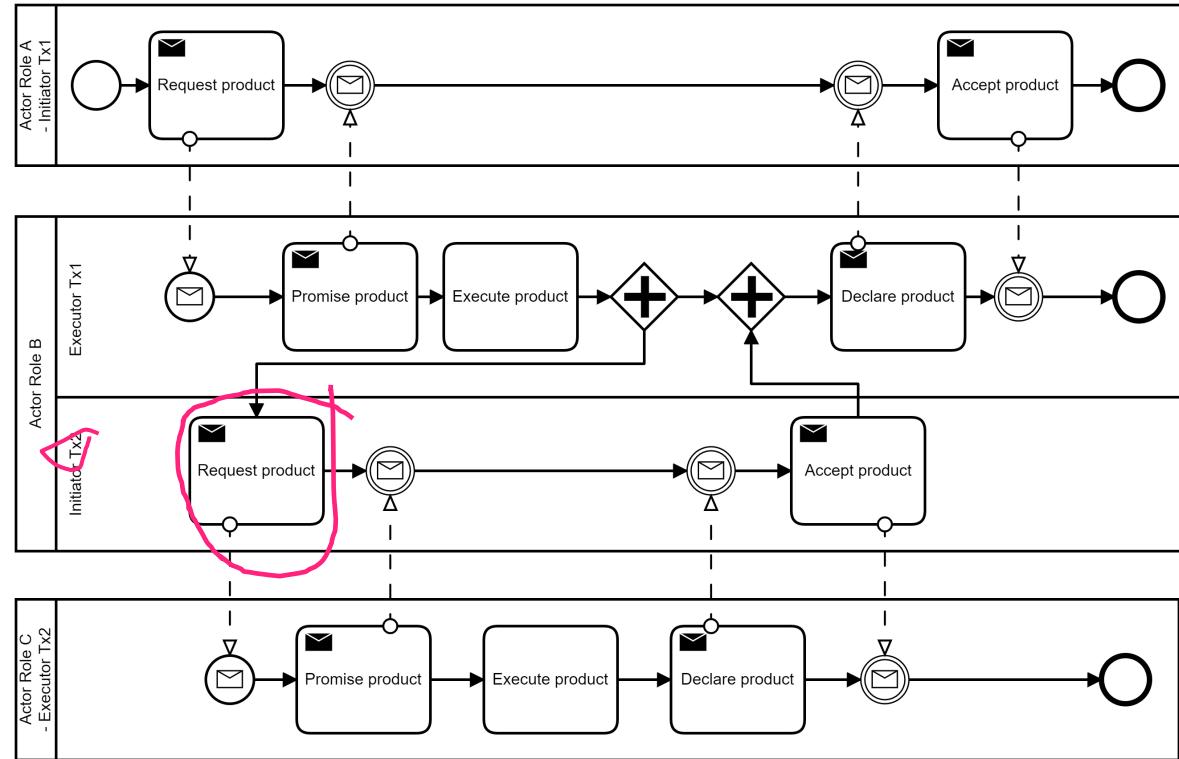


Transactions dependency – Request after Promise

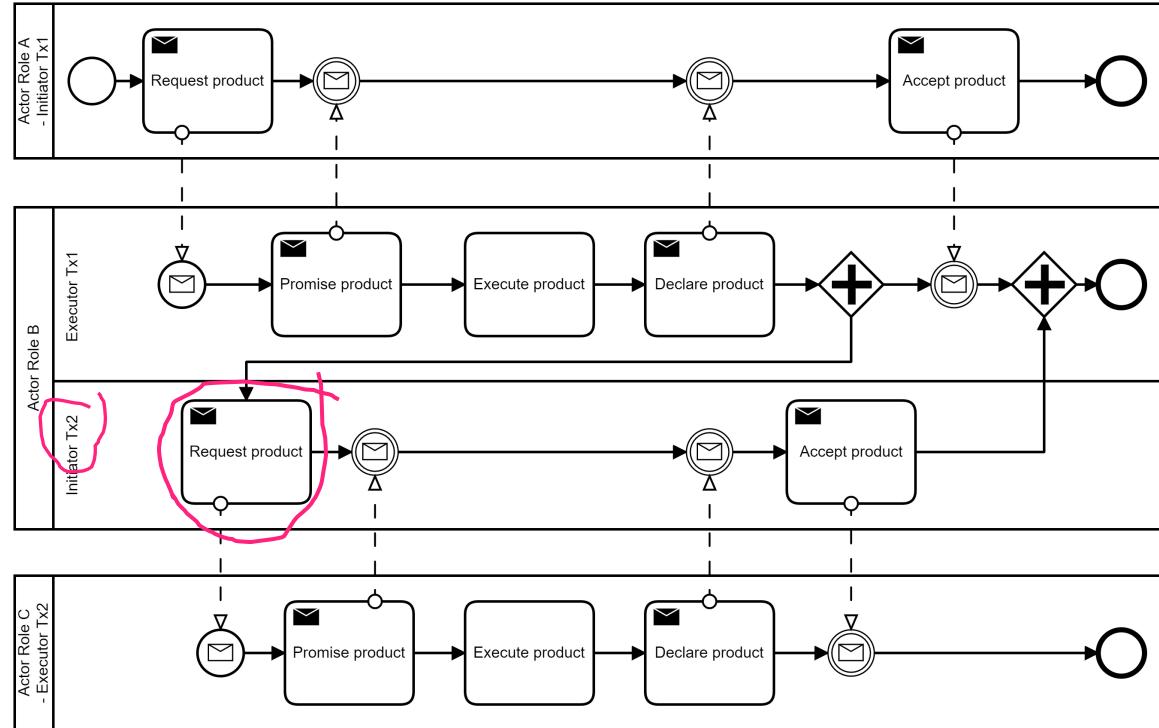




Transactions dependency – Request after Execute

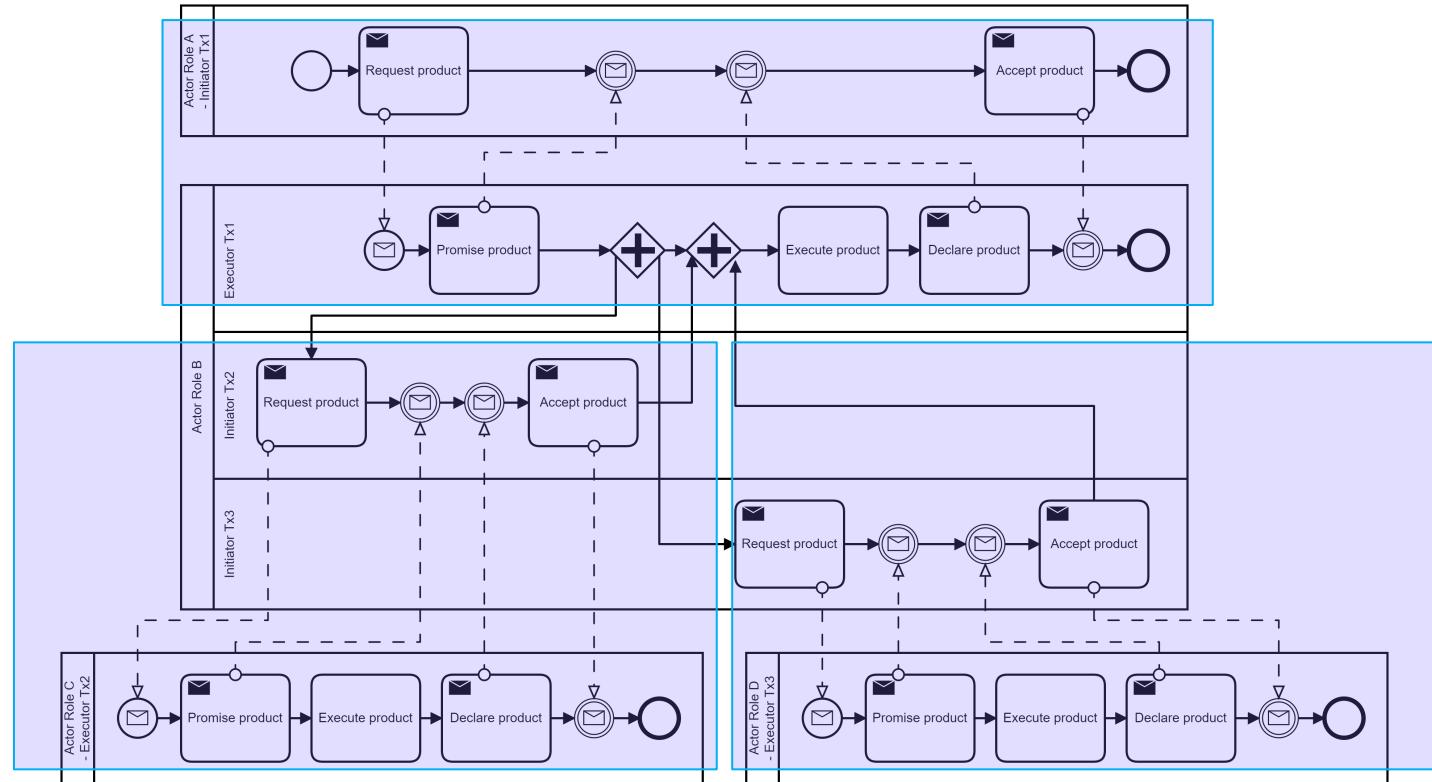


Transactions dependency – Request after Declare





RaP composition pattern concerning 3 business transactions where TK2 and TK3 are performed simultaneously.





<https://github.com/SemantifyingBPMN/SemantifyingBPMN>

```
target % java -jar SemantifyingBPMN-4.0.1.jar
```

The usage of SemantifyingBPMN is the following.

```
SemantifyingBPMN-4.0.1 --actors <filename> --tpt <filename> --tkdepend <filename> --output-file-txt <filename> --output-file-bpmn <filename>
Credits: Sérgio Guerreiro (2022) (github: https://github.com/SemantifyingBPMN/SemantifyingBPMN)
```

where the parameters are,

--actors: is a csv file with the list of actor roles and is mandatory. Composed of 2 fields, in each line, with actor role name and description:
(e.g.: A01 – Customer ; The role that initiates the business process).

--tpt: is a csv file with the Transactor Product Table and is mandatory. Composed of 6 fields, in each line, with TK name, TK description, Actor role initiator, Actor role executor, Product kind , Product kind description:

(e.g.: TK01; Sale completing ; A01 – Customer ; A02 – Dispatcher ; PK01 ; [Product] is sold).

--tkdepend: is a csv file with the dependencies matrix N*N transactions and is mandatory. Composed of Strings with dependencies: **RaP** = Request after Promise pattern, **RaE** = Request after Execution, **RaD** = Request after Declare pattern.

(e.g.:

```
    ; TK01 ; TK02 ; TK03 ; TK04
TK01 ;           ;           ;
TK02 ; RaP      ;           ;
TK03 ;           ; RaE     ;
TK04 ;           ;           ; RaE
)
```

--tkview: is a mandatory csv file with view definition for each transaction per line, acceptable values are: **HappyFlow** |
HappyFlowAndDeclinationsAndRejections | **Complete** | **Custom** | **CustomHappyFlowOnly**. Default value is HappyFlow.

The Custom value accepts extra detail for each transaction step, even empty ones.

(e.g.

```
    TransactionKind ; View   ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare      ;
Decision Accept ; Accept  ; Reject   ; Evaluate Rejection ; Stop
    TK01 ; HappyFlow
    TK02 ; HappyFlowAndDeclinationsAndRejections
    TK03 ; Custom      ; Pedido    ;           ; Executa ; how to decide      ; Produce ; Here it is      ;
Valida resultado;           ; not ok ; decide reject  ; ok
    TK04 ; Complete
    TK05 ;
)
```

--businessObjects: is a csv file with the list of business object for each transaxction step.

(e.g.

```
    TransactionKind ; TransactionStep ; BusinessObject
    TK01 ; Request   ; Order
    TK02 ; Promise   ; Receipt
)
```

--simplify on|off: create a BPMN model without communications for simplification purposes only. Optional, default is off.

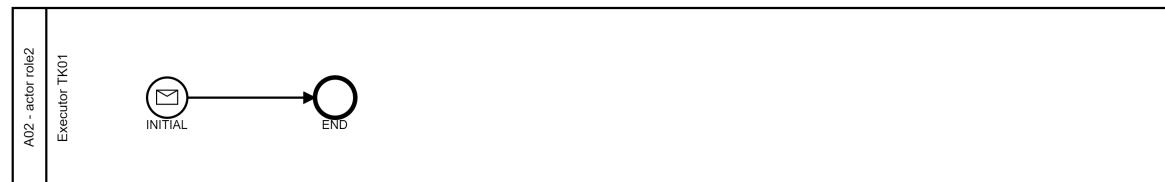
--engine camunda|... : create a BPMN model ready for CAMUNDA execution. Optional.

--output-file-txt: is a file to store the model in txt format. Optional.

--output-file-bpmn: is a file to store the BPMN model. Optional.

BPMNSemantifying tool OUTCOME

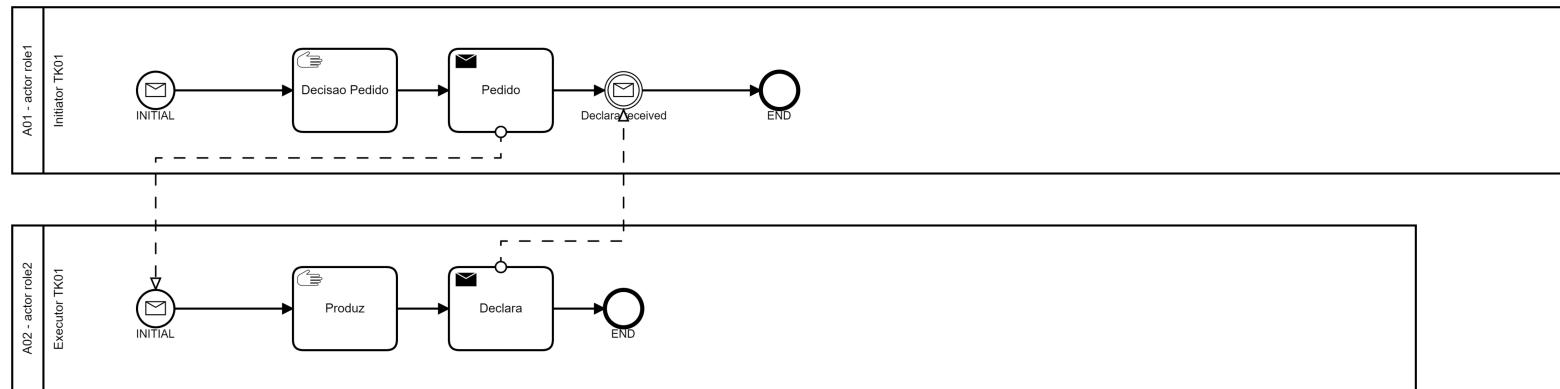
```
root@0736c3e84db2:~# more actors
A01 - actor role1 ; Budget transfer solicitor.
A02 - actor role2 ; Executor of the first validation and creator of a budget transfer proposal.
root@0736c3e84db2:~# more tpt
TK01;Soliciting budget change; A01 - actor role1 ; A02 - actor role2 ; PK01 ; [budget] has been
changed
root@0736c3e84db2:~# more tkdepend
      ; TK01
TK01;
```



BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more actors
A01 - actor role1 ; Budget transfer solicitor.
A02 - actor role2 ; Executor of the first validation and creator of a budget transfer proposal.
root@0736c3e84db2:~# more tpt
TK01;Soliciting budget change; A01 - actor role1 ; A02 - actor role2 ; PK01 ; [budget] has been
changed
root@0736c3e84db2:~# more tkdepend
    ; TK01
TK01;
```

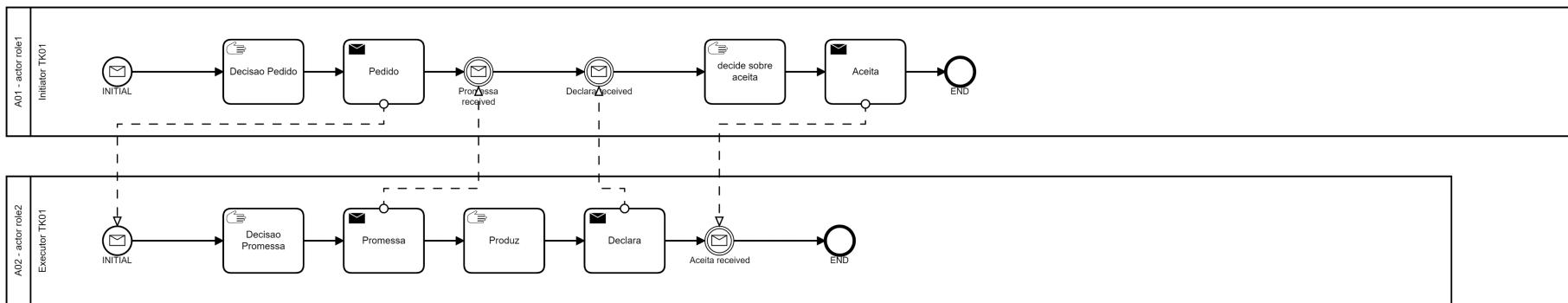
```
root@0736c3e84db2:~# more tkview2
TransactionKind ; View ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare ; Decision Accept ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01 ; CustomHappyFlowOnly ; Decisao Pedido ; Pedido ; ; ; ; Produz ; Declara ; ; ; ; ;
```



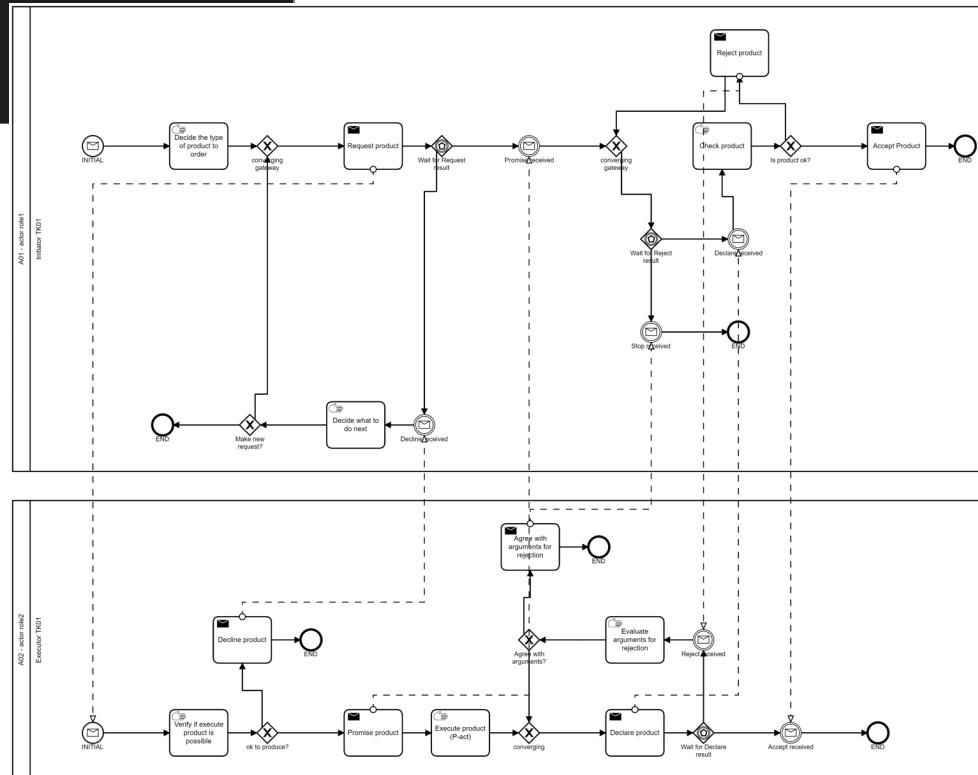
BPMNSemantifying tool OUTCOME

```
- □ ×  
root@0736c3e84db2:~# more actors  
A01 - actor role1 ; Budget transfer solicitor.  
A02 - actor role2 ; Executor of the first validation and creator of a budget transfer proposal.  
root@0736c3e84db2:~# more tpt  
TK01;Soliciting budget change; A01 - actor role1 ; A02 - actor role2 ; PK01 ; [budget] has been  
changed  
root@0736c3e84db2:~# more tkdepend  
    ; TK01  
TK01;
```

```
- □ ×  
root@0736c3e84db2:~# more tkview1  
TransactionKind ; View ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare ; Decision Accept ; Accept ; Reject ; Evaluate Rejection ; Stop  
TK01 ; CustomHappyFlowOnly ; Decisao Pedido ; Pedndo ; Decisao Promessa ; Promessa; ; Produz; Declara ; decide sobre aceita ; Aceita; ; ;
```



```
root@0736c3e84db2:~# more actors
A01 - actor role1 ; Budget transfer solicitor.
A02 - actor role2 ; Executor of the first validation and creator of a budget transfer proposal.
root@0736c3e84db2:~# more tpt
TK01;Soliciting budget change; A01 - actor role1 ; A02 - actor role2 ; PK01 ; [budget] has been
changed
root@0736c3e84db2:~# more tkdepend
    ; TK01
TK01;
```



BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more tkview6
```

```
TransactionKind ; View ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare ; Decision Accept ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01          ; Complete ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;
```

```
root@0736c3e84db2:~# more actors
```

A01 - actor role1 ; Budget transfer solicitor.

A02 - actor role2 ; Executor of the first validation and creator of a bud

```
root@0736c3e84db2:~# more tpt
```

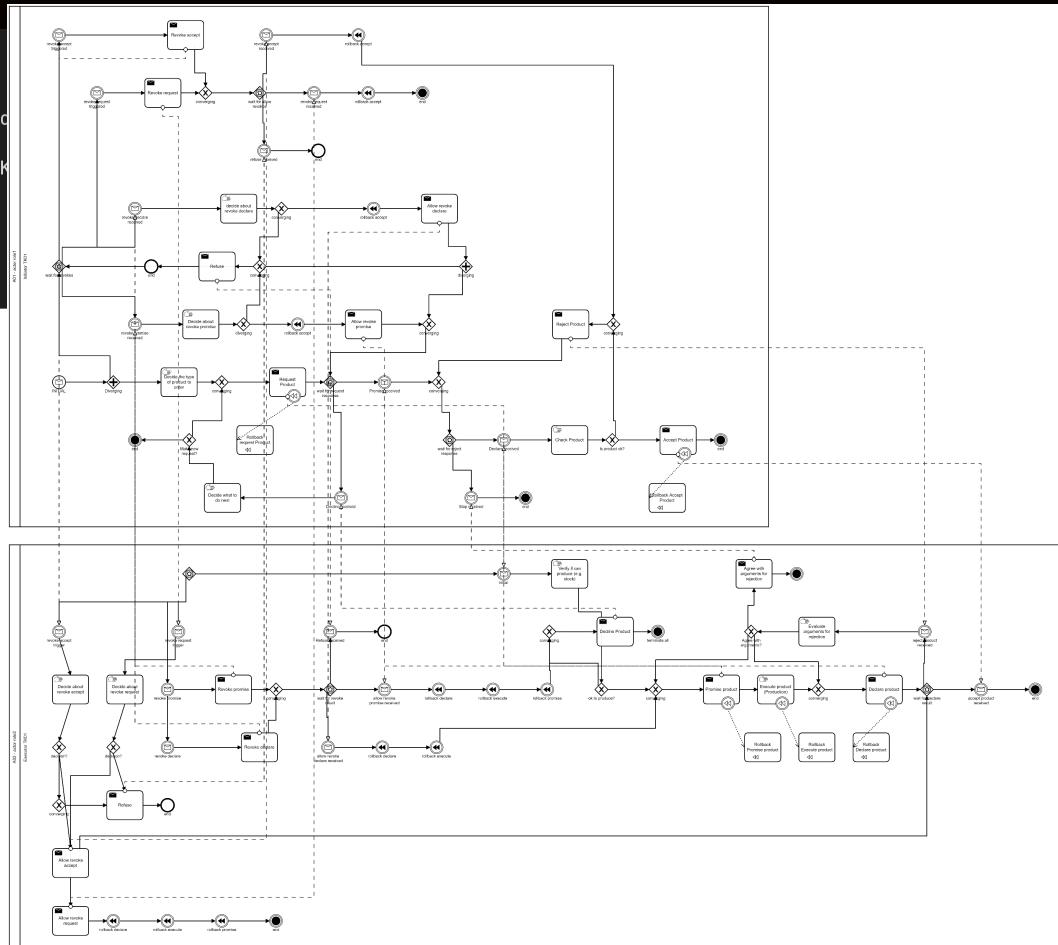
TK01;Soliciting budget change; A01 - actor role1 ; A02 - actor role2 ; PK

changed

```
root@0736c3e84db2:~# more tkdepend
```

; TK01

TK01;



BPMNSemantifying tool OUTCOME

```
root@0736c3e84db2:~# more actors4
```

A01 - SOC Department ; Budget transfer solicitor.
A02 - SPFP ; Executor of the first validation and creator of a budget transfer proposal.
A03 - SPFP Coordinator : Executor of the second validation of the budget transfer.
A04 - Financial Department Director ; Executor of the third validation of the budget transfer.
A05 - Governing Board Representative ; Executor of the fourth validation of the budget transfer.

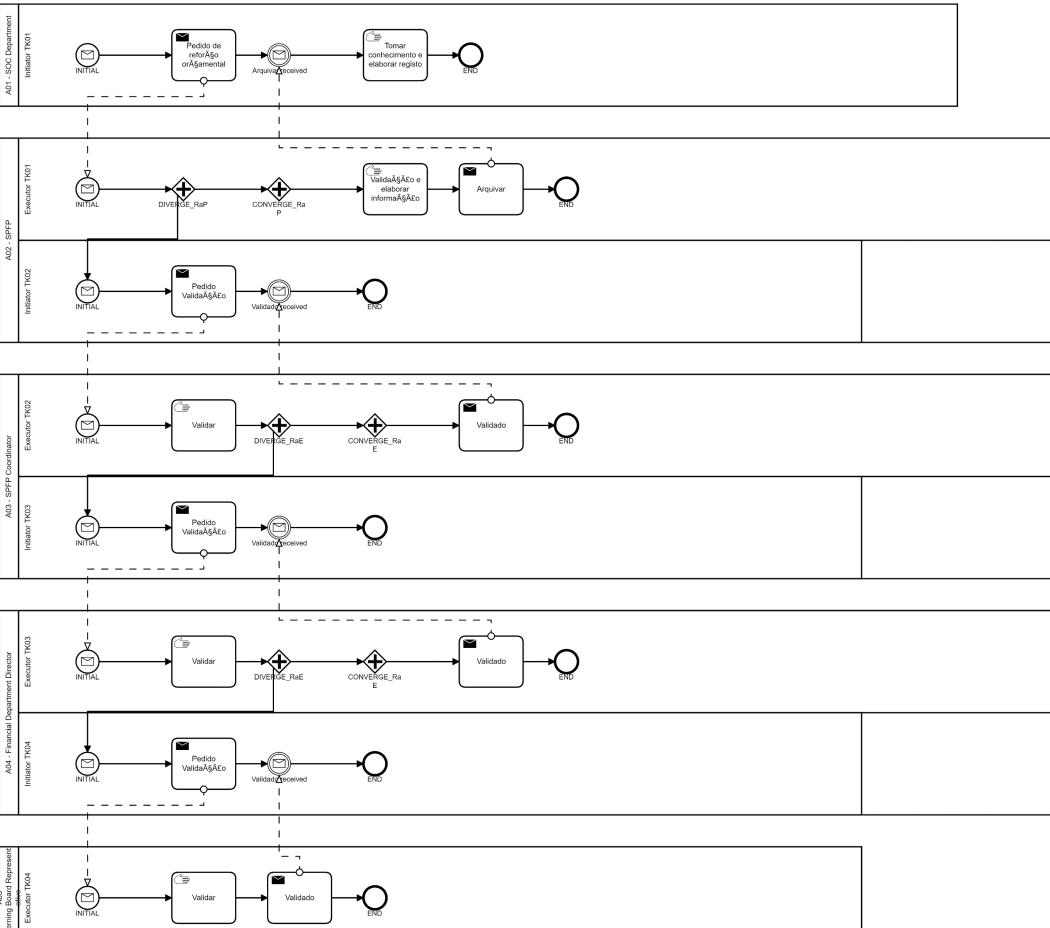
```
root@0736c3e84db2:~# more tktp4
```

TK01;Soliciting budget change; A01 - SOC Department ; A02 - SPFP ; PK01 ; [budget] has been changed
TK02;Validating second-level budget change; A02 - SPFP ; A03 - SPFP Coordinator; PK02 ; [budget change] has been validated by second level
TK03;Validating third-level budget change;A03 - SPFP Coordinator;A04 - Financial Department Director; PK03 ; [budget change] has been valid
TK04;Validating fourth-level budget change;A04 - Financial Department Director;A05 - Governing Board Representative ; PK04 ; [budget change]

```
root@0736c3e84db2:~# more tkdepend4
```

; TK01; TK02;TK03;TK04
TK01; ; ;
TK02; RaP ; ;
TK03; ; RaE ; ;
TK04; ; ;RaE ;

BPMN Semantifying tool OUTCOME



```
root@0736c3e84db2:~# more tkview4
```

TransactionKind : View ; Request Decision; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare ; Decision Accept ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01 ; CustomHappyFlowOnly ; Pedido de reforço orçamental ; ; ; ; ; ;
TK02 ; CustomHappyFlowOnly ; Pedido Validação ; ; ; ; ; ;
TK03 ; CustomHappyFlowOnly ; Pedido Validação ; ; ; ; ; ;
TK04 ; CustomHappyFlowOnly ; Pedido Validação ; ; ; ; ; ;

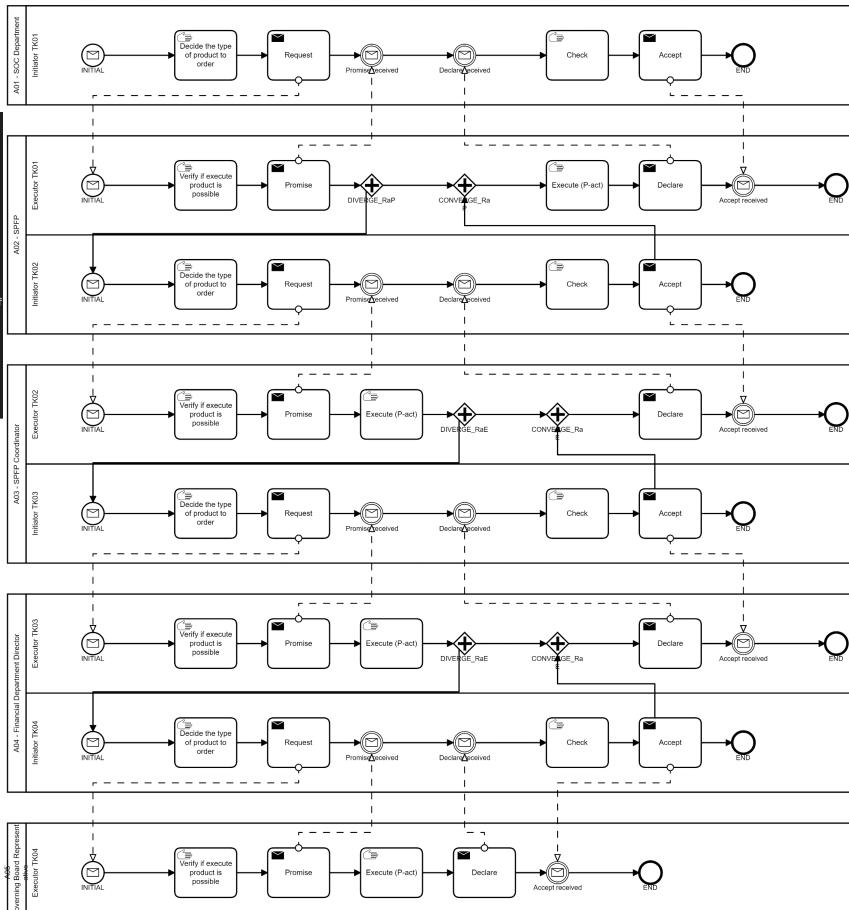
```

root@0736c3e84db2:~# more actors4
A01 - SOC Department ; Budget transfer solicitor.
A02 - SPFP ; Executor of the first validation and creator of a budget transfer proposal.
A03 - SPFP Coordinator : Executor of the second validation of the budget transfer.
A04 - Financial Department Director ; Executor of the third validation of the budget transfer.
A05 - Governing Board Representative ; Executor of the fourth validation of the budget transfer.

root@0736c3e84db2:~# more tkdepend4
; TK01; TK02;TK03;TK04
TK01; ;
TK02; RaP ;
TK03; RaE ;
TK04; ;RaE ;

TK01; TK02;TK03;TK04
TK02;Validating second-level budget change; A02 - SPFP ; A03 - SPFP Coordinator; PK02 ; [budget change] has been validated by second level
TK03;Validating third-level budget change;A03 - SPFP Coordinator;A04 - Financial Department Director; PK03 ; [budget change] has been validated by third level
TK04;Validating fourth-level budget change;A04 - Financial Department Director;A05 - Governing Board Representative; PK04 ; [budget change] has been validated by f

```



BPMNSemantifying tool OUTCOME

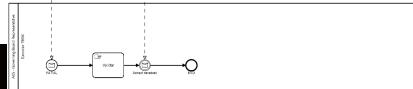
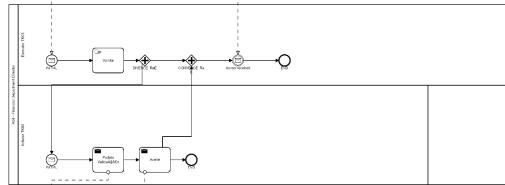
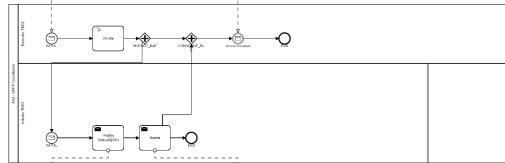
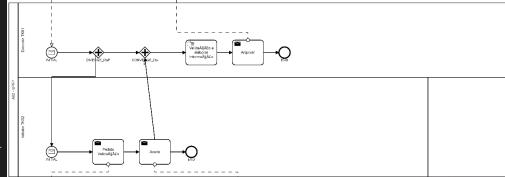
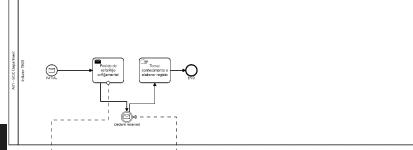
TransactionKind	View	Request Decision	Request	Promise Decision	Promise	Decline	After Decline Decision	Execute	Decision	Declare	Decision Accept	Accept	Reject	Evaluate Rejection	Stop
TK01	HappyFlow		Pedido de reforço orçamental						Validação e elaborar informação	Arquivar		Tomar conhecimento e elaborar registo			
TK02	HappyFlow		Pedido Validação						Validar	Validado					
TK03	HappyFlow		Pedido Validação						Validar	Validado					
TK04	HappyFlow		Pedido Validação						Validar	Validado					

BPMNSemantifying tool OUTCOME

```
root@0736c3eb4db2:~# more actors4
A01 - SOC Department ; Budget transfer solicitor.
A02 - SPFP ; Executor of the first validation and creator of a budget transfer proposal.
A03 - SPFP Coordinator ; Executor of the second validation of the budget transfer.
A04 - Financial Department Director ; Executor of the third validation of the budget transfer.
A05 - Governing Board Representative ; Executor of the fourth validation of the budget transfer.
```

```
root@0736:3e84db2:# more tpt4
TK01;Soliciting budget change;A01 - SOC Department ; A02 - SPFP ; PK01 ; [budget] has been changed
TK02;Validating second-level budget change;A02 - SPFP ; A03 - SPFP Coordinator ; PK02 ; [budget change] has been validated by second level
TK03;Validating third-level budget change;A03 - SPFP Coordinator;A04 - Financial Department Director; PK03 ; [budget change] has been validated by third level
TK04;Validating fourth-level budget change;A04 - Financial Department Director;A05 - Governing Board Representative; PK04 ; [budget change] has been validated by fourth level
```

```
root@0736c3e84bd2:~# more tkdepend4  
; TK01; TK02;TK03;TK04  
TK01; ; ; ;  
TK02; RaP ; ; ;  
TK03; ; RaE ; ;  
TK04; ; ;RaE ;
```

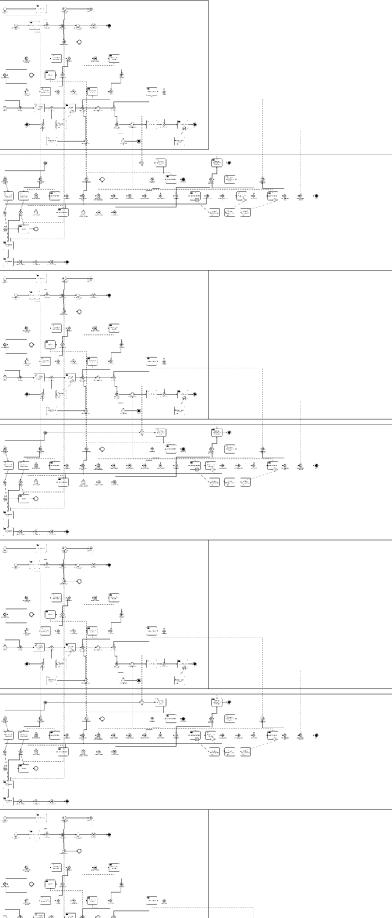


```
root@0736c3e84db2:# more tkview6
TransactionKind ; View ; Request Decision ; Request ; Promise Decision ; Promise ; Decline ; After Decline Decision ; Execute ; Declare ; Decision Accept ; Accept ; Reject ; Evaluate Rejection ; Stop
TK01 ; Custom; ; Pedido de reforço orçamental ; ; ; ; ; ; ; ; ; ; ; ; ; ;
TK02 ; Custom; ; Pedido Validação ; ; ; ; ; ; ; ; ; ; ; ; ; ;
TK03 ; Custom; ; Pedido Validação ; ; ; ; ; ; ; ; ; ; ; ; ; ;
TK04 ; Custom; ; Pedido Validação ; ; ; ; ; ; ; ; ; ; ; ; ; ;
```

```
root@736c3e84db2:~# more actors4
A01 - SOC Department ; Budget transfer solicitor.
A02 - SPPF ; Executor of the first validation and creator of a budget transfer proposal.
A03 - SPPF Coordinator ; Executor of the second validation of the budget transfer.
A04 - Financial Department Director ; Executor of the third validation of the budget transfer.
A05 - Governing Board Representative ; Executor of the fourth validation of the budget transfer.

root@736c3e84db2:~# more tpt4
TK01;Soliciting budget change;A01 - SOC Department ; A02 - SPPF ; PK01 ; [budget] has been changed
TK02;Validating second-level budget change;A02 - SPPF ; A03 - SPPF Coordinator; PK02 ; [budget change] has been validated by second level
TK03;Validating third-level budget change;A03 - SPPF Coordinator;A04 - Financial Department Director; PK03 ; [budget change] has been validated by third level
TK04;Validating fourth-level budget change;A04 - Financial Department Director;A05 - Governing Board Representative; PK04 ; [budget change] has been validated by fourth level

root@736c3e84db2:~# more tkdepend4
; TK01; TK02;TK03;TK04
TK01; : ; ;
TK02; RaB ; ;
TK03; ; RaE ;
TK04; ; ;RaB ;
```



This BPMN XML file produced in some numbers:

8460 lines • 26 send tasks • 76 exclusive gateways •

14 parallel gateways • 48 intermediateThrowEvent •

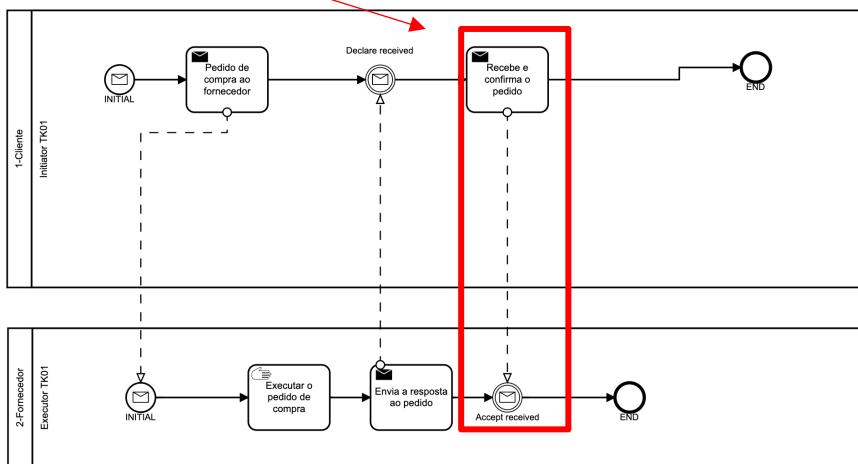
40 manualTasks • 560 BPMN edges • 68 messageFlows •

430 flowNodeRefs ...

Visualizing the patterns



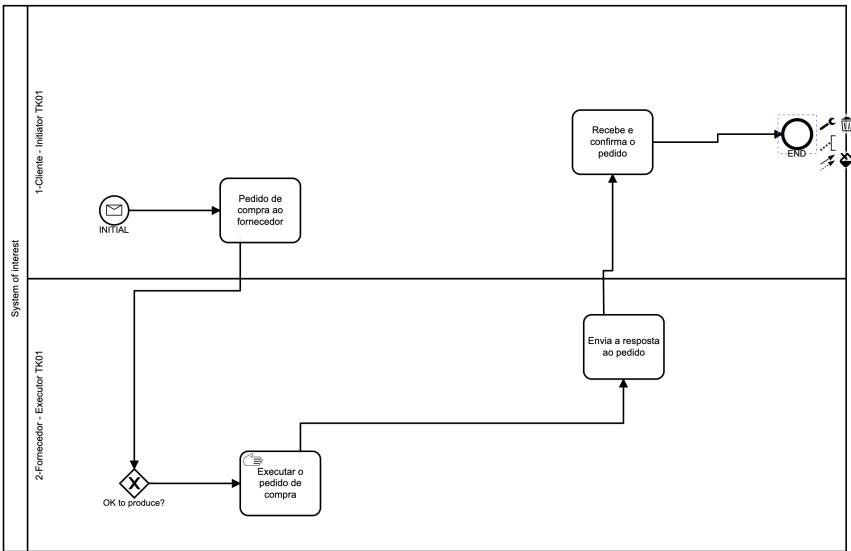
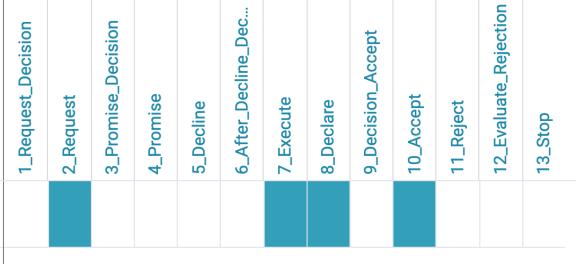
BPMN PRODUCED IN MULTIPLE POOLS





BPMN PRODUCED IN MULTIPLE LANES

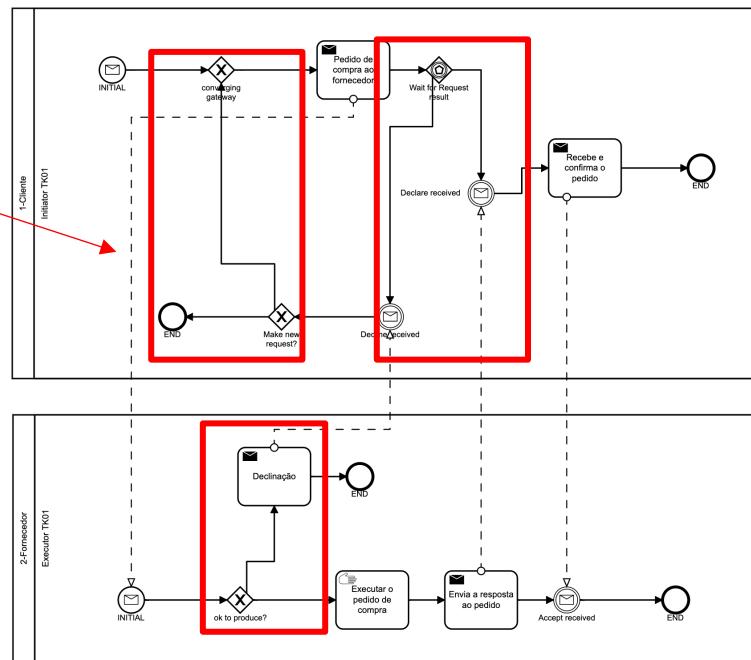
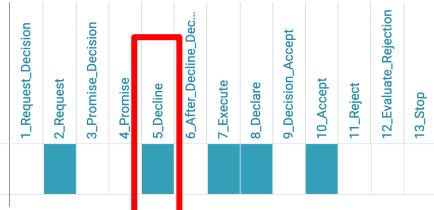
TK1: From (1-Cliente) to (2-Fornecedor)





ADDING EXCEPTIONS: REQUEST DECLINATION USING POOLS

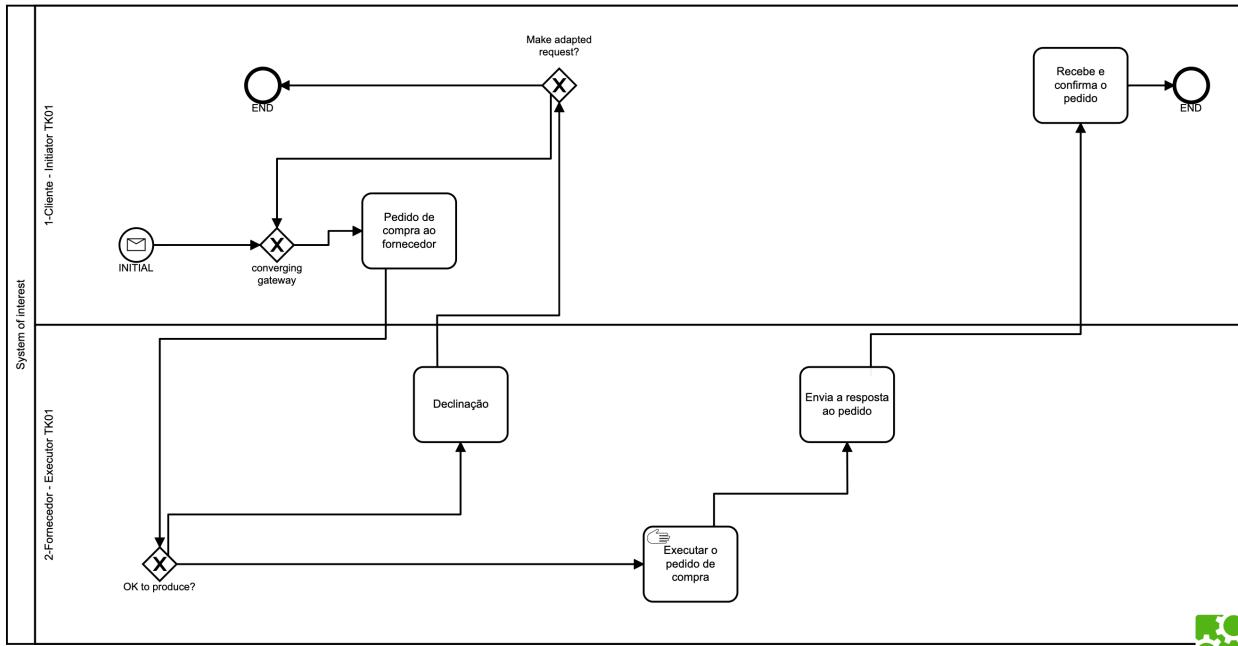
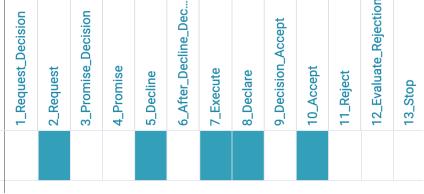
TK1: From (1-Cliente) to (2-Fornecedor)





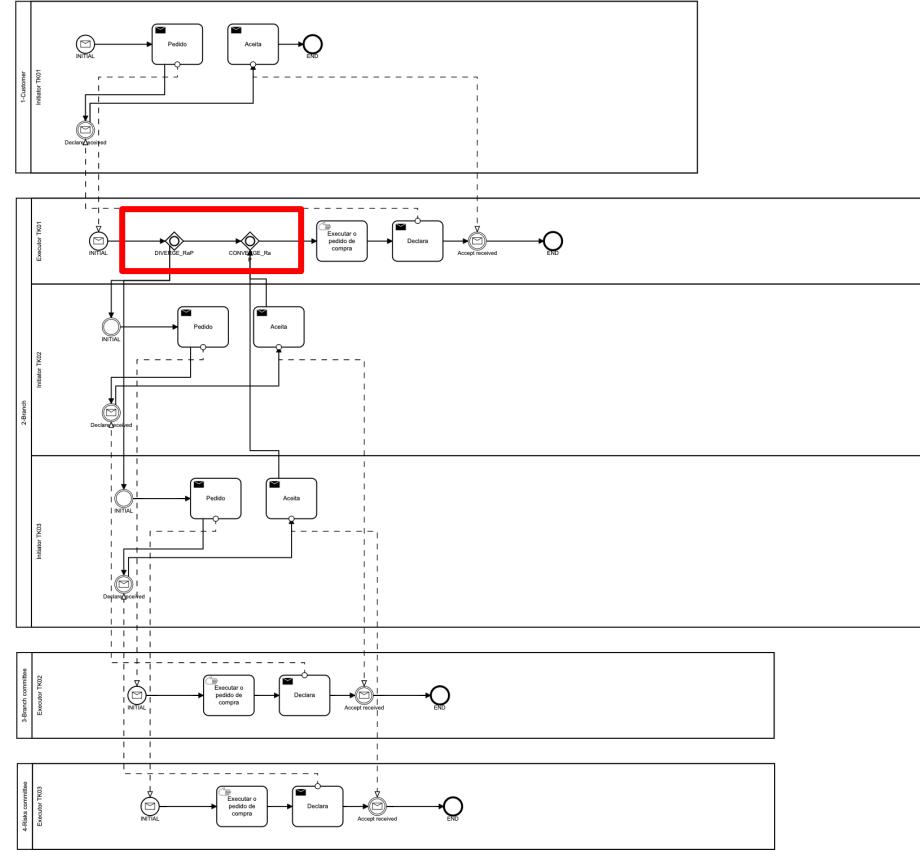
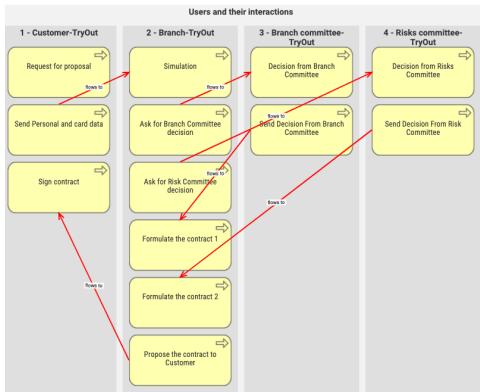
ADDING EXCEPTIONS: REQUEST DECLINATION USING LANES

TK1: From (1-Cliente) to (2-Fornecedor)

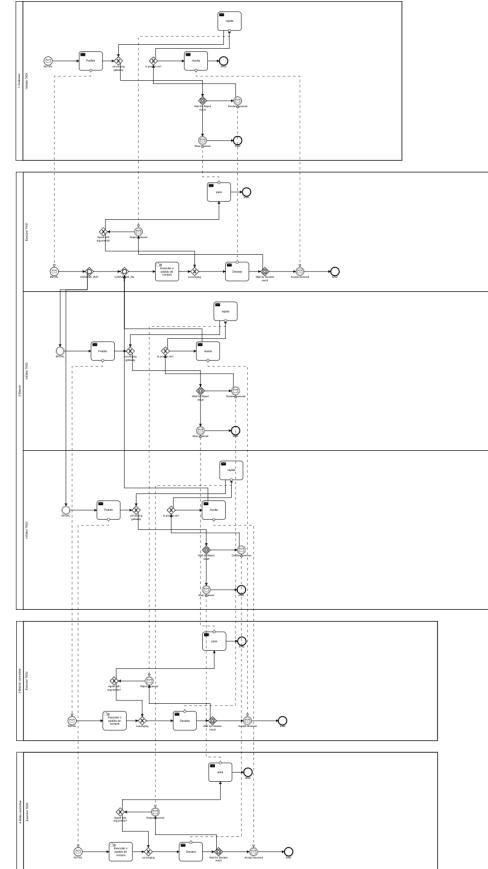
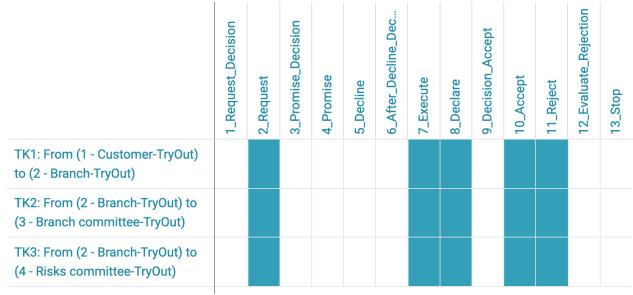




PRODUCING A NETWORK OF 4 ACTORS + 3 TKS USING POOLS



ADDING REJECTION TO THE NETWORK OF 4 ACTORS + 3 TKS USING POOLS



Q&A





TÉCNICO LISBOA

Enterprise Integration

Application Programming Interfaces (API) Management

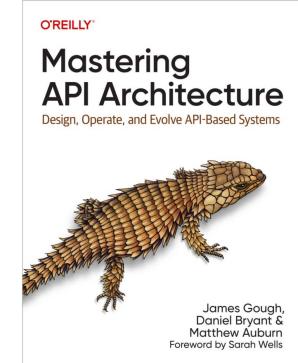
Prof. Sérgio Guerreiro

Sergio.guerreiro@tecnico.ulisboa.pt

Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

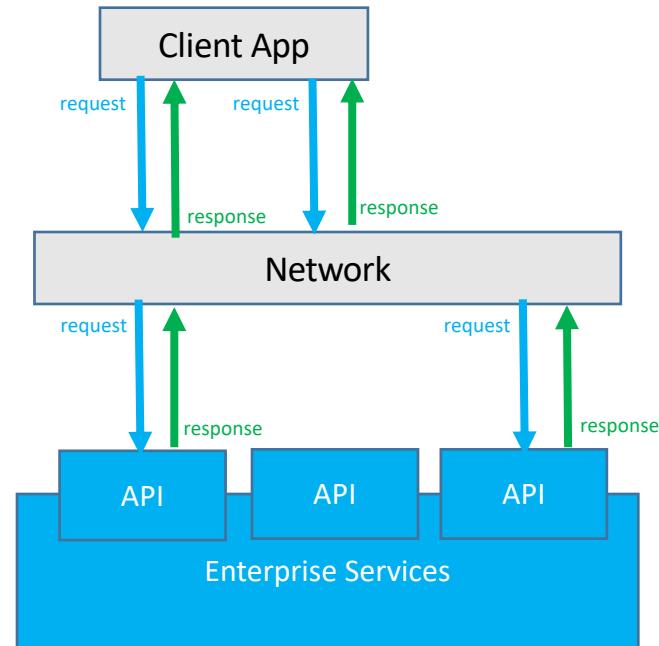
Gough, J., Bryant, D., Auburn, M. (2023).
Mastering API Architecture: Design, Operate,
and Evolve API-Based Systems, O'Reilly Media,
Inc., ISBN: 9781492090632



Application Programming Interfaces (APIs)

APIs are *by definition* **interfaces** to be used as entry points to reusable software entities with well-defined contracts.

They are not independent software entities; they are instead packaged with the software libraries, frameworks, or Web services, that offer them.



A broader API definition

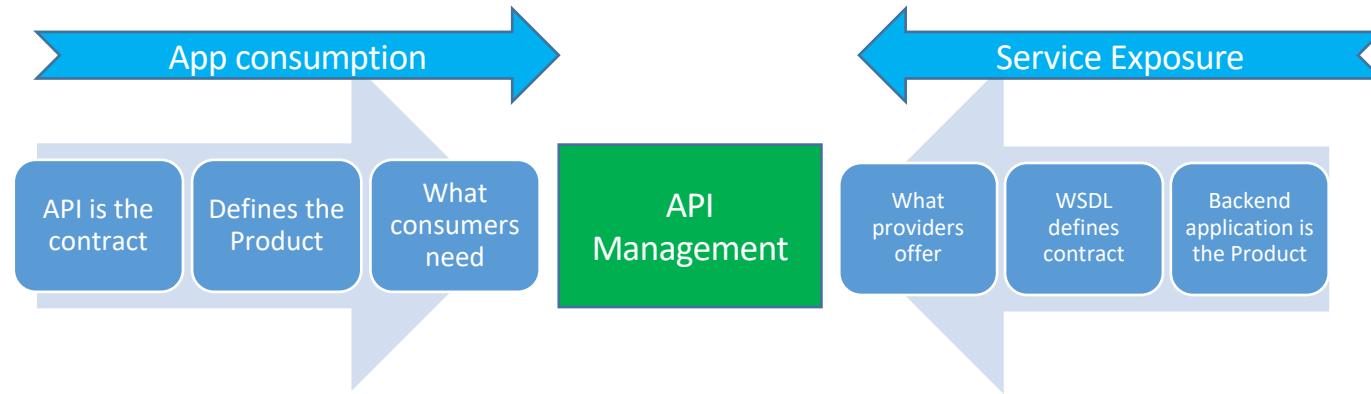
- An API represents an abstraction of the underlying implementation
- An API is represented by a specification that introduces types. Developers can understand the specifications and use tooling to generate code in multiple languages to implement an API consumer (software that consumes an API).
- An API has defined semantics or behavior to effectively model the exchange of information.
- Effective API design enables extension to customers or third parties for a business integration.
- ***in process API invocation***: e.g., a java RMI call is handled by the same process from which the call was made
- ***out of process API invocation***: e.g., a .NET application invoking an external REST-like API using an HTTP library is handled by an additional external process other than the process from which the call was made.

Application Programming Interfaces (APIs)

- Due to their omnipresence and evolution, APIs **greatly impact software development**. Understanding, mitigating, and leveraging the impact of APIs and API evolution on software development is necessary to design and use software APIs
- APIs evolve for various reasons, such as increasing complexity, and continuous change. However, due to their nature as a **connection point** between software modules, **API evolution is not without side effects**
- On the one hand, as predicted by **Lehman, continuing change** means that API developers must determine ways to keep their APIs useful, cutting edge, and competitive with other pieces of software and API users must adapt to these API changes and new API releases.
- On the other hand, **conservation of familiarity**, or existing API usages, constrain the evolution of an API to avoid breaking changes while improving the API (*i.e.*, security or performance improvements). The evolution of APIs therefore involves a balancing **act of constant improvement and maintaining existing functionality**. Maintaining existing functionality requires in-depth knowledge of use cases and architectural foresight and flexibility, while keeping up with rapid release cycles requires modifications to user applications as well as learning about new APIs and changes to existing APIs.

Differences from SOA

- From an architectural perspective they are similar: “*a logical representation of a repeatable activity that has a specific outcome, the usual notion of a service*”
- But the focus is different:
 - APIs were focused in simplifying the consumption, developer centric, human-readable contract
 - SOA focus in shielding the client from back-office applications changes



API trends

- **New tools and techniques** typically seek to help with API evolution by **resolving problems** that it can cause for API users (e.g., API migration tools) or to help **reduce the development** burden on API developer (e.g., automatic API documentation tools).
- Empirical studies related to API evolution typically employ large data, case studies, or user studies to provide evidence of existing problems, the impacts of API evolution, or potential solutions to existing problems. These problems typically centre around the impacts of API evolution on **API usability** and **API maintainability**.

State-of-the-art Solutions to Existing API Evolution Challenges

Challenge	Proposed State-of-the-Art Solution
Dealing with breaking API changes	Document acceptable usages [179] API bundles [103] Automated API migration [24, 57] Extracting migration knowledge from clients [86, 158]
Improving API usability	Local interaction patterns [65] Usability patterns [202] API selection criteria [117] Digital assistants [16] Automated API tips [175] Automated documentation [2, 71, 98, 149, 171] Example mining [58, 116, 168] API recommendation algorithms [12, 99, 125, 190]
Reducing API misuses	Misuse detectors [4, 5, 9]

Open Challenges in API Research

Challenge types	Paper types	Challenges
Existing challenges	New tools and techniques	<i>EC-1</i> Combining textual merging with syntactic and semantic approaches [109]
		<i>EC-2</i> Providing a commercially viable API migration solution [20, 34]
		<i>EC-3</i> Incorporating domain-specific information into tools [109]
		<i>EC-4</i> Using systematic evaluation methods in empirical evaluations [145]
		<i>EC-5</i> Producing more specific and less abstract theories [102]
		<i>EC-6</i> Reducing the variability of software API studies [102]
		<i>EC-7</i> Finding input examples for API migration through examples [148]
		<i>EC-8</i> Improving the granularity of API migration approaches [148]
		<i>EC-9</i> Validation and correction of API migration edit scripts [148]
		<i>EC-10</i> More tools to help with Web APIs [180]
		<i>EC-11</i> Using existing library API research as stepping stones for Web APIs [180]
		<i>EC-12</i> Combining both API side learning with client side learning [158]
	Empirical studies	<i>EC-13</i> Dealing with out-of-vocabulary problems [24]
		<i>EC-14</i> Defining best fit APIs [192]
		<i>EC-15</i> Automatically identifying factors that drive API changes [60, 70, 191]
		<i>EC-16</i> Dealing with API semantics and dependencies [5]
		<i>EC-17</i> Deploying bug fixes to multiple API versions [160]

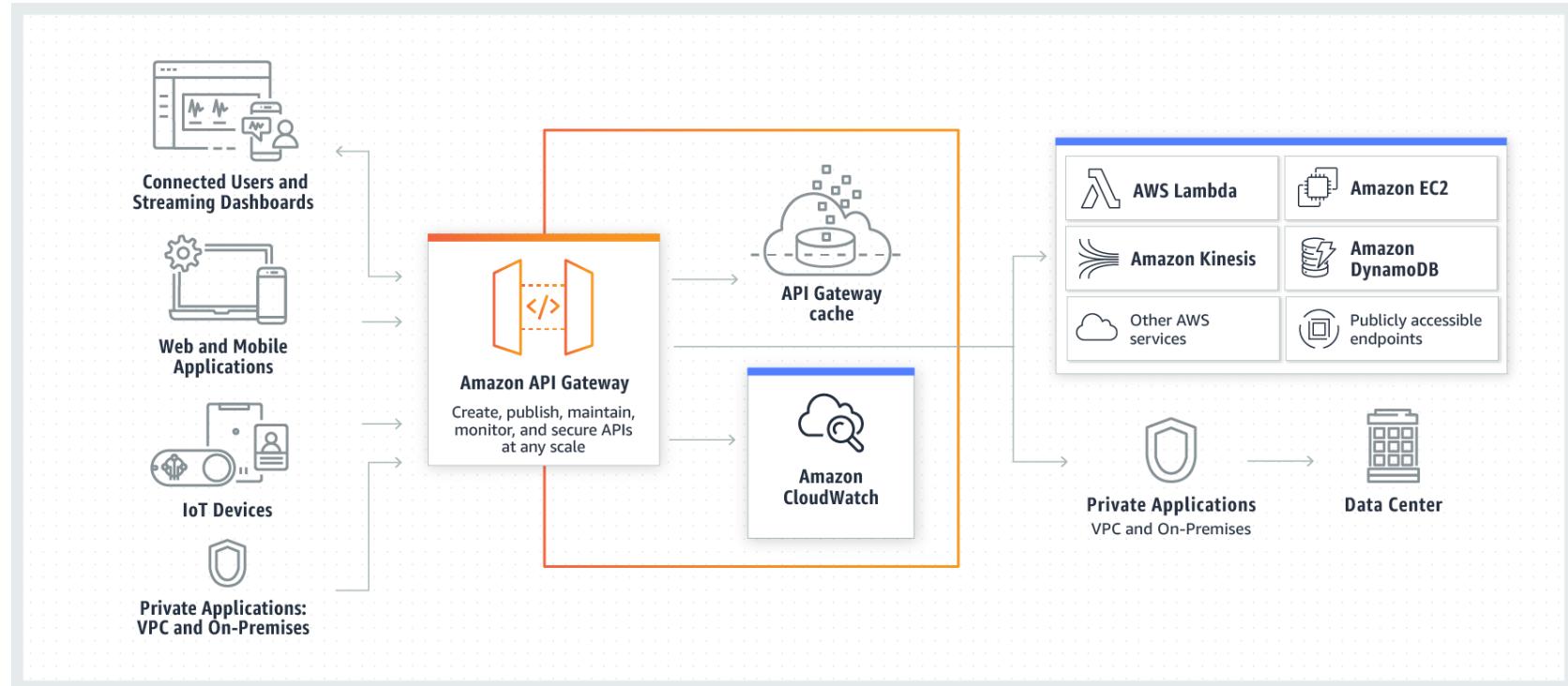
Open Challenges in API Research

Unsolved Challenges	New tools and techniques	<i>UC-1</i> Using uniform benchmarks for API tool evaluation
		<i>UC-2</i> Supporting the context sensitivity of API migration tools
		<i>UC-3</i> Improving performance of API tooling to allow user adoption
		<i>UC-4</i> Dealing with fuzzy and ambiguous developer intent
		<i>UC-5</i> Reducing the knowledge gap between API users and developers
		<i>UC-6</i> Tools that mine usage data help API developers improve APIs
		<i>UC-7</i> Keeping API users in the loop for API recommendation systems
		<i>UC-8</i> Generalizing API tools to languages other than Java
		<i>UC-9</i> Tools to help API developers deal with API migration, not just users
		<i>UC-10</i> Reducing API misuse from the API development side
	Empirical studies	<i>UC-11</i> Understanding the coupling between APIs and programming languages
		<i>UC-12</i> Determining API migration and API recommendation impacts
		<i>UC-13</i> Generalizing API empirical studies to languages other than Java
		<i>UC-14</i> Comparing the evolution of various APIs
	Datasets	<i>UC-15</i> Creating large-scale API migration and recommendation datasets

What is API management?

“...is a set of tools and services that enable developers and companies to build, analyze, operate, and scale APIs in secure environments. API management can be delivered on-premises, through the cloud, or using a hybrid on-premises – SaaS.”

AWS API management landscape



Important features of API management tools

- **API ACCESS CONTROL:** APIs should be built using access controls, commonly known as authentication and authorization, that grant users permission to access certain systems, resources, or information.
- **API PROTECTION:** API protections include API keys for identification, API secrets, and application authorization tokens that can be verified.
- **API CREATION AND DESIGN:** APIs allow web applications to interact with other applications. You can create and define different types of APIs such as RESTful APIs and WebSocket APIs.
- **SUPPORT FOR HYBRID MODELS:** A RESTful API is a group of resources and methods, or endpoints, that leverage an HTTP request type. A WebSocket API maintains a persistent connection between connected clients.
- **HIGH PERFORMANCE:** Highly performant APIs depend on code, the separation of functionalities, and on underlying data structure and data architecture.
- **CUSTOMIZABLE DEVELOPER PORTAL:** API developer portals connect API publishers with API subscribers. They enable self-service API publishing and allow potential API customers to easily discover APIs they can use.

Representation State Transfer (REST)

“...is a set of architectural constraints, most commonly applied using HTTP as the underlying transport protocol...”

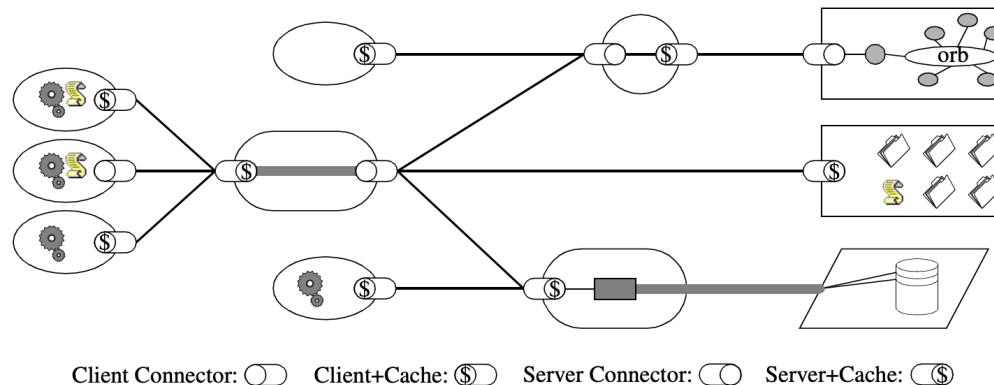


Figure 5-8. REST

Representation State Transfer (REST)

To be considered RESTful your API must ensure that:

1. A **producer-to-consumer** interaction is modeled where the producer models resources the consumer can interact with.
2. Requests from producer to consumer are **stateless**, meaning that the producer doesn't cache details of a previous request. In order to **build up a chain of requests** on a given resource, the consumer must send any required information to the producer for processing.
3. Requests are **cachable**, meaning the producer can provide hints to the consumer where this is appropriate. In HTTP this is often provided in information contained in the header.
4. A **uniform interface** is transmitted to the consumer.
5. It is a **layered system**, abstracting away the complexity of systems sitting behind the REST interface. For example, the consumer should not know or care if they are interacting with a database or other services.

REST data elements I

The key abstraction of information in REST is a **resource**.

Any information that can be named can be a resource:

- a document or image,
- a temporal service (e.g. “today’s weather in Los Angeles”),
- a collection of other resources,
- a non-virtual object (e.g. a person), and so on.

In other words, any concept that might be the target of an author’s hypertext reference must fit within the definition of a resource.

A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time.

Table 5-1. REST Data Elements

Data Element	Modern Web Examples
resource	the intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
resource metadata	source link, alternates, vary
control data	if-modified-since, cache-control

REST data elements II

REST components perform actions on a resource by using a **representation** to capture the current or intended state of that resource and transferring that representation between components.

A representation is a sequence of bytes, plus representation metadata to describe those bytes.

Other commonly used but less precise names for a representation include: document, file, and HTTP message entity, instance, or variant.

Table 5-1. REST Data Elements

Data Element	Modern Web Examples
resource	the intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
resource metadata	source link, alternates, vary
control data	if-modified-since, cache-control

REST data elements III

A representation is a sequence of bytes, plus **representation metadata** to describe those bytes.

The data format of a representation is known as a **media type**.

Some media types are intended for automated processing, some are intended to be rendered for viewing by a user, and a few are capable of both.

Composite media types can be used to enclose multiple representations in a single message.

Response messages may include both representation metadata and **resource metadata**: information about the resource that is not specific to the supplied representation.

Table 5-1. REST Data Elements

Data Element	Modern Web Examples
resource	the intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
resource metadata	source link, alternates, vary
control data	if-modified-since, cache-control

REST data elements III

Control data defines the **purpose of a message between components**, such as the action being requested or the meaning of a response. It is also used to parameterize requests and override the default behavior of some connecting elements. For example, cache behavior can be modified by control data included in the request or response message.

A representation can be included in a message and processed by the recipient according to the **control data** of the message and the nature of the media type.

Table 5-1. REST Data Elements

Data Element	Modern Web Examples
resource	the intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
resource metadata	source link, alternates, vary
control data	if-modified-since, cache-control

REST over HTTP

An **HTTP request** is:

- A verb (aka method), most of the time one of [GET](#), [POST](#), [PUT](#), [DELETE](#) or [PATCH](#)
- A URL
- Headers (key-value pairs)
- Optionally a body (aka payload, data)

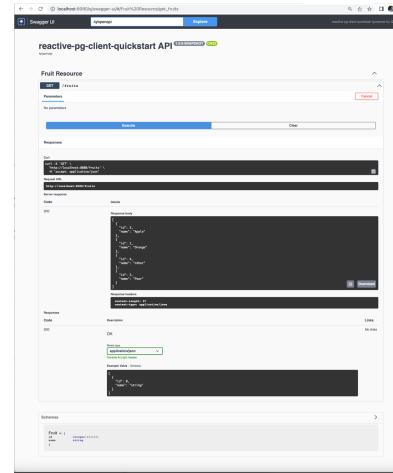
An **HTTP response** is:

- A status, most of the time one of [2xx \(successful\)](#), [4xx \(client error\)](#) or [5xx \(server error\)](#)
- Headers (key-value pairs)
- A body (aka payload, data)

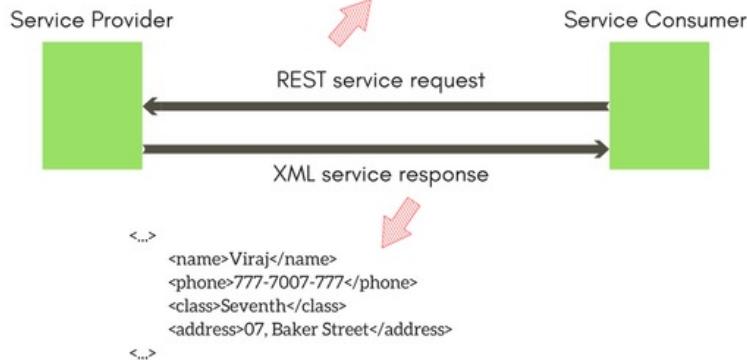
HTTP verbs characteristics:

- Verbs that have a body: POST, PUT, PATCH
- Verbs that must be safe (i.e. that mustn't modify resources): GET
- Verbs that must be idempotent (i.e. that mustn't affect resources again when run multiple times): GET (nullipotent), PUT, DELETE

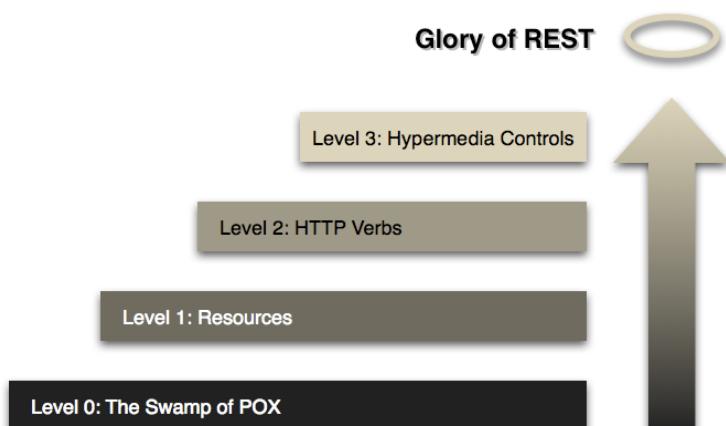
	body	safe	idempotent
2 GET	x	✓	✓
3 POST	✓	x	x
4 PUT	✓	x	✓
5 DELETE	x	x	✓
6 PATCH	✓	x	x



GET <http://www.studytonight.com/student/studentRollno/07>



Richardson maturity heuristics



- Level 0: The starting point for the model is using HTTP as a transport system for remote interactions, but without using any of the mechanisms of the web. Essentially, using HTTP as a tunneling mechanism for remote interaction mechanism, usually based on Remote Procedure Invocation.
- Level 1 – introduce **Resources**: rather than making all our requests to a singular service endpoint, start talking to individual resources.
 - tackles the question of handling complexity by using divide and conquer, breaking a large service endpoint down into multiple resources.
- Level 2 - using the **HTTP verbs**: as closely as possible to how they are used in HTTP itself.
 - introduces a standard set of verbs so that we handle similar situations in the same way, removing unnecessary variation.
- Level 3 - Hypermedia Controls **introduces HATEOAS** (Hypertext As The Engine Of Application State).
 - Level 3 introduces discoverability, providing a way of making a protocol more self-documenting.

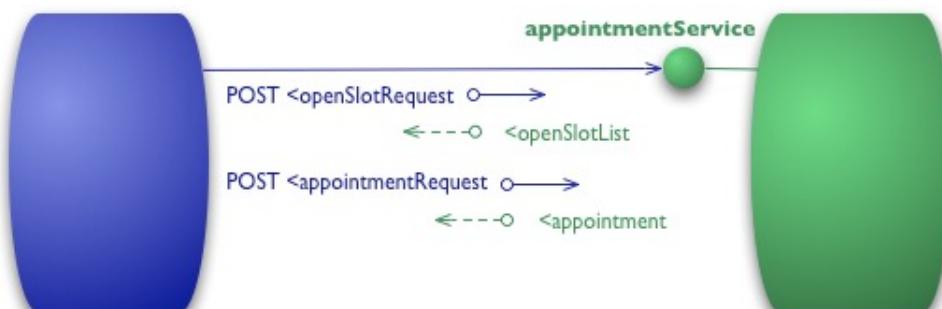
Level 0

Request:

```
POST /appointmentService HTTP/1.1
[various other headers]
<openSlotRequest date = "2010-01-04" doctor = "mjones"/>
```

Response:

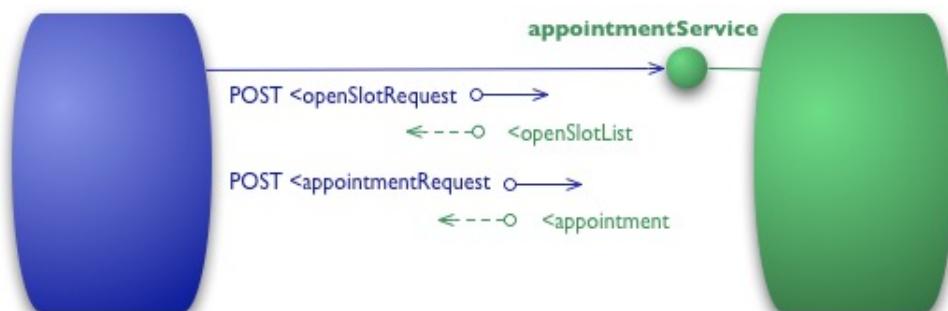
```
HTTP/1.1 200 OK
[various headers]
<openSlotList>
  <slot start = "1400" end = "1450">
    <doctor id = "mjones"/>
  </slot>
  <slot start = "1600" end = "1650">
    <doctor id = "mjones"/>
  </slot>
</openSlotList>
```



Level 0

Request:

```
POST /appointmentService HTTP/1.1
[various other headers]
<appointmentRequest>
    <slot doctor = "mjones" start = "1400" end = "1450"/>
    <patient id = "jsmith"/>
</appointmentRequest>
```



Response:

HTTP/1.1 200 OK

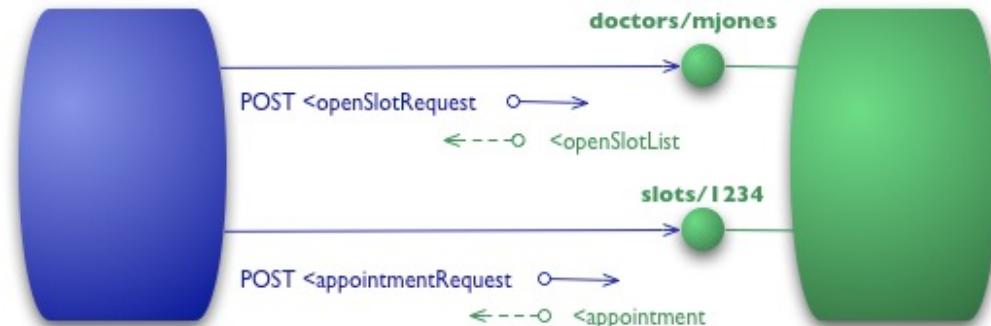
[various headers]

```
<appointment>
    <slot doctor = "mjones" start = "1400" end = "1450"/>
    <patient id = "jsmith"/>
</appointment>
```

Level 1

Request:

```
POST /doctors/mjones HTTP/1.1
[various other headers]
<openSlotRequest date = "2010-01-04"/>
```



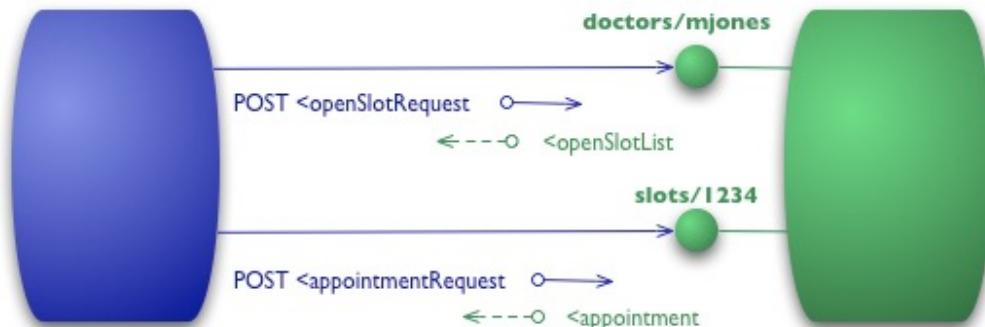
Response:

```
HTTP/1.1 200 OK
[various headers]
<openSlotList>
    <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
    <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

Level 1

Request:

```
POST /slots/1234 HTTP/1.1
[various other headers]
<appointmentRequest>
    <patient id = "jsmith"/>
</appointmentRequest>
```



Response:

```
HTTP/1.1 200 OK
[various headers]
<appointment>
    <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
    <patient id = "jsmith"/>
</appointment>
```

Level 2

Request:

```
GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1
```

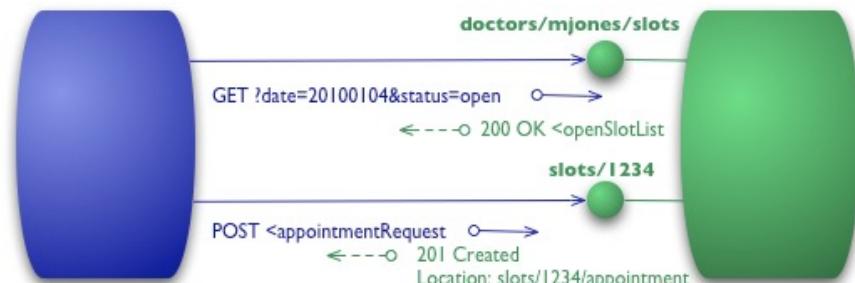
Host: royalhope.nhs.uk

Response:

HTTP/1.1 200 OK

[various headers]

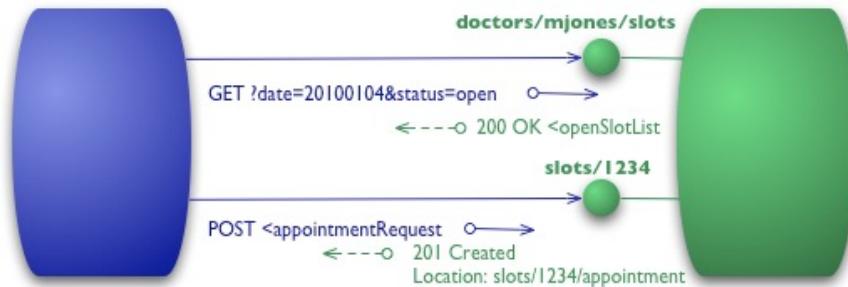
```
<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```



Level 2

Request:

```
POST /slots/1234 HTTP/1.1
[various other headers]
<appointmentRequest>
    <patient id = "jsmith"/>
</appointmentRequest>
```



Response:

```
HTTP/1.1 201 Created Location: slots/1234/appointment
[various headers]
<appointment>
    <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
    <patient id = "jsmith"/>
</appointment>
```

Level 3

Request:

GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1

Host: royalhope.nhs.uk

Response:

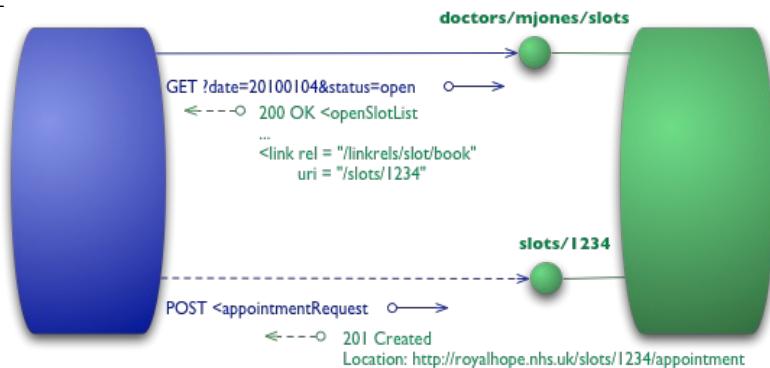
HTTP/1.1 200 OK

[various headers]

<openSlotList>

```

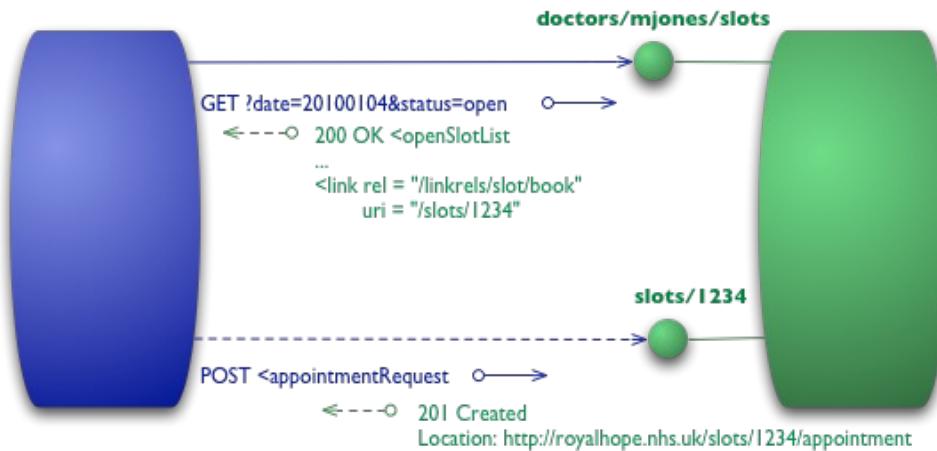
<slot id = "1234" doctor = "mjones" start = "1400" end = "1450">
    <link rel = "/linkrels/slot/book" uri = "/slots/1234"/>
</slot>
<slot id = "5678" doctor = "mjones" start = "1600" end = "1650">
    <link rel = "/linkrels/slot/book" uri = "/slots/5678"/>
</slot>
</openSlotList>
```



Level 3

Request:

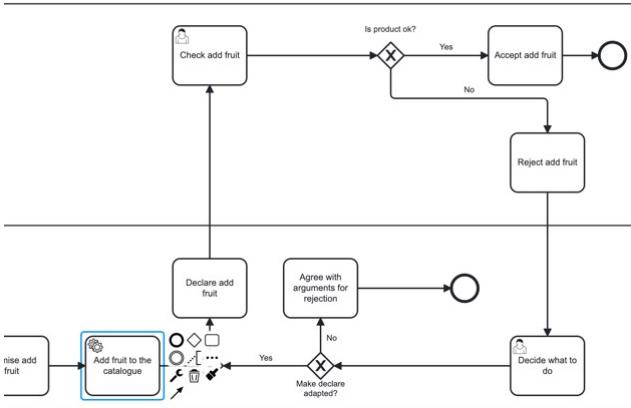
```
POST /slots/1234 HTTP/1.1
[various other headers]
<appointmentRequest>
    <patient id = "jsmith"/>
</appointmentRequest>
```



Response:

```
HTTP/1.1 201 Created Location: http://royalhope.nhs.uk/slots/1234/appointment
[various headers]
<appointment>
    <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
    <patient id = "jsmith"/>
    <link rel = "/linkrels/appointment/cancel" uri = "/slots/1234/appointment"/>
    <link rel = "/linkrels/appointment/addTest" uri = "/slots/1234/appointment/tests"/>
    <link rel = "self" uri = "/slots/1234/appointment"/>
    <link rel = "/linkrels/appointment/changeTime" uri = "/doctors/mjones/slots?date=20100104&status=open"/>
    <link rel = "/linkrels/appointment/updateContactInfo" uri = "/patients/jsmith/contactInfo"/>
    <link rel = "/linkrels/help" uri = "/help/appointment"/>
</appointment>
```

Camunda REST example



```

curl -X 'POST' \
'http://ec2-54-234-15-238.compute-1.amazonaws.com:8080/fruits' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "id": 0,
  "name": "sd"
}'
  
```

Asynchronous continuations >

Inputs	+
Connector inputs	+
headers	
Local variable name	headers
Assignment type	Map
Map entries	
Content-Type	
Key	Content-Type
Value	application/json
accept	
Key	accept
Value	application/json
method	
Local variable name	method
Assignment type	String or expression
Value	POST
Start typing "\${!}" to create an expression.	
payload	
Local variable name	payload
Assignment type	String or expression
Value	{ "id": 0, "name": "\${NewFruit}" }
Start typing "\${!}" to create an expression.	
url	
Local variable name	url
Assignment type	String or expression
Value	http://ec2-54-234-15-238.compute-1.amazonaws.com:8080/fruits
Start typing "\${!}" to create an expression.	

headers

verb

payload

URL

Open API

- REST APIs are technological implementation of services and do not account with contracting issues
- Usually an additional document, or wiki documents the API usage. However, those are hard to maintain and testing is not automating
- “*The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to HTTP APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic*”

<https://swagger.io/specification/>

Swagger

Swagger has evolved into one of the most widely used open source tool sets for developing APIs with rich support for the OpenAPI Specification, AsyncAPI specification, JSON Schema and more.

<https://swagger.io/tools/open-source/>

The contract of an API encompasses:

- **Title**
- **Endpoint**
- **Method**
- **URL parameters**
- **Message payload**
- **Header parameters**
- **Response code**
- **Error code**
- **A sample request and response**
- **Tutorials and walkthrough**
- **Service-level agreement**



Design

Design APIs in a powerful editor which visually renders your OpenAPI or AsyncAPI definition and provides real-time error feedback.

Swagger Editor



Build

Build and enable consumption of your API by generating server stubs and client SDKs with minimal plumbing.

Swagger Codegen



Document

Automatically generate documentation from your OpenAPI definition for visual interaction, and easier consumption.

Swagger UI

Swagger for Everyone

Swagger open source and pro tools have helped millions of API developers, teams, and organizations deliver great APIs.

Swagger

- Example of swagger embedded with Quarkus to facilitate the documentation and testing of a Project

localhost:8080/q/swagger-ui/#/Fruit%20Resource/post_fruits__stock_

Swagger UI qopenapi Explore reactive-mysql-transactioncontrol-client-quickstart2023 API 1.0.0-SNAPSHOT (GAS)

Fruit Resource

GET /fruits

GET /fruits/{id}

DELETE /fruits/{id}

PUT /fruits/{id}/{name}

POST /fruits/{stock}

Parameters

Name	Description
stock <small>required</small>	Integer(1)@min(1)
name	string

Request body application/json

```
{
  "id": 0,
  "name": "nova fruta"
}
```

Responses

Curl

```
curl -X POST \
  "http://localhost:8080/fruits/56" \
  -H "Content-Type: application/json" \
  -H "User-Agent: curl/8.1.4" \
  -d "{
    \"id\": 0,
    \"name\": \"nova fruta\"
  }"
```

Request URL http://localhost:8080/fruits/56

Server response

Code	Details
201	Created
	Response headers Content-length: 0 Location: http://localhost:8080/fruits/56
Responses	
Code	Description
200	OK
	Media type application/json
	Controls Accept header

Schemas

Fruit v 0

```

id   Integer(1)@min(1)
name string

```

GraphQL

- Is an API technology developed by Facebook to address some of the challenges faced with RESTful APIs, particularly in complex applications where multiple data sources need to be queried.
- Allows the client to request only the specific data it needs, using a **declarative query language**, reducing over-fetching and under fetching issues common with RESTful APIs.
- Provides a flexible and efficient way for front-end applications, making it a choice for applications with complex data requirements and a variety of front-end interfaces. It allows developers to define their queries precisely, resulting in fewer network requests and less data transfer.
- However, comes with additional complexity compared to RESTful APIs and requires more careful management of server-side schema

Query	Response
<pre>{ hero { name height mass } }</pre>	<pre>{ "hero": { "name": "Luke Skywalker", "height": 1.72, "mass": 77 } }</pre>

gRemote Procedure Call (gRPC)

- Developed by Google, is a high-performance, open-source API framework that uses HTTP2 for transport and protocol buffers as data format. gRPC is designed to provide efficient communication between microservices, making it highly suitable for blockchain backends that require fast and reliable communication with front-end clients.
- One of the advantage is its ability to support **bi-directional streaming**, which is particularly useful for real-time applications where low latency is critical. Unlike RESTful APIs, gRPC allows the server to push updates to the client, making it suitable for applications that require continuous data streams.
- gRPC's performance and efficiency make it an excellent option for projects that demand low latency and high data throughput, such as real-time trading platforms or blockchain monitoring services.
- However, gRPC is more complex to implement than REST or GraphQL and may require more setup, particularly for handling serialization and managing HTTP2 connections. Despite this, its performance benefits are invaluable for projects with stringent speed and reliability requirements.

a. S. C. Spotlight, "API Types and Protocols" <https://www.browserstack.com/guide/angular-vs-react-vs-vue>, 2024.

MuleSoft, "What is a gRPC API and how does it work?" <https://www.mulesoft.com/api-university/what-grpc-api-and-how-does-it-work>, 2024.

D. Strmecki, "REST vs. GraphQL vs. gRPC - Which API to Choose?" <https://www.baeldung.com/rest-vs-graphql-vs-grpc>, 2024.

Enterprise Integration

API Gateway Systems

Prof. Sérgio Guerreiro

Sergio.guerreiro@tecnico.ulisboa.pt

Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

Gateway (in general terms)

- According to Gartner¹, a gateway in computer networking:

“gateway converts information, data, or other communications from one protocol or format to another”

- Moreover, it must allow communication in both directions and maintain the connections. A gateway can also be seen as a “supervised entry point” through which messages (carries of information/knowledge) can enter a particular environment undergiven conditions and rules applied.

¹ <https://www.gartner.com/en/information-technology/glossary/gateway>

API Gateway System (in general terms)

Provides or integrates with third-party gateways for runtime management, security, policy enforcement, throttling, operational control and usage monitoring for APIs.

Specific:

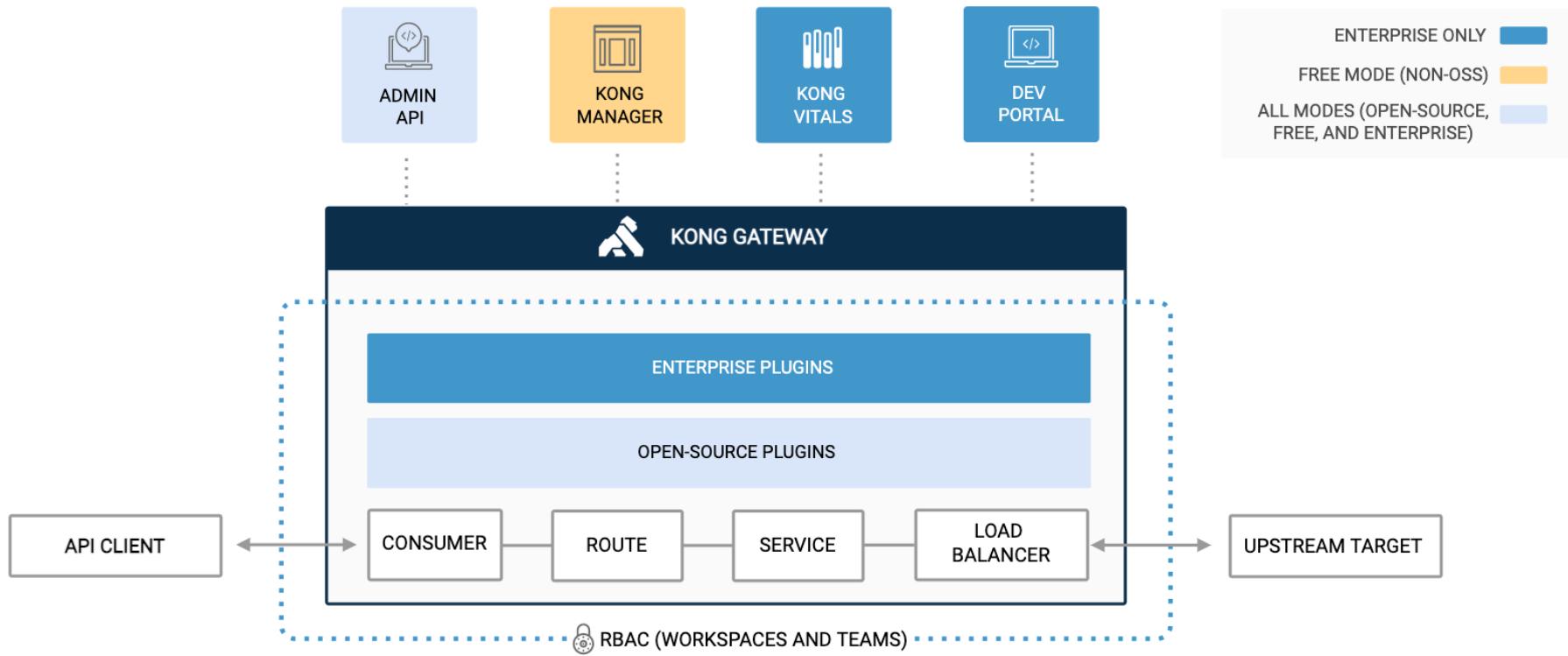
- Kong Gateway, is a commercial version of its open-source API gateway based on NGINX and OpenResty.
- Kong Gateway works as a reverse proxy¹ that manage, configure, and route requests to other APIs.
- Kong Gateway runs in front of any RESTful API and can be extended through modules and plugins. It's designed to run on decentralized architectures, including hybrid-cloud and multi-cloud deployments.

¹ A reverse proxy server is a type of proxy server that typically sits behind the firewall in a private network and directs client requests to the appropriate backend server

Advantages of an API Gateway System

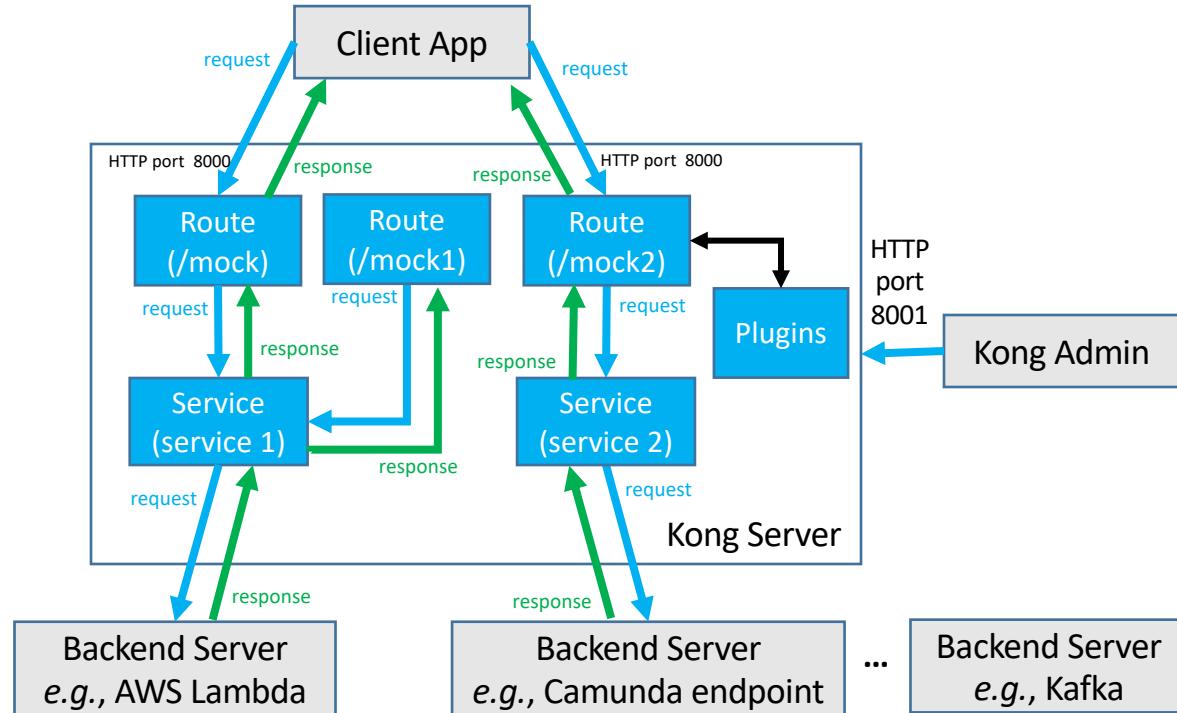
- Enable new consumers without coding or configuration
 - Interface and Volume Scoping
 - Routing
- Keep unwanted people & robots out of your systems
 - Authorization and Authentication
 - Threat Protection
- Understand what is happening with your APIs
 - Discovery, Usage, and Concerns
 - History

API gateway core functionalities



Kong main concepts

- **Service**: is the name Kong uses to refer to the upstream APIs and microservices it manages
- **Route**: specify how (and if) requests are sent to their Services after reaching Kong
- A single Service can have multiple Routes.
- **Plugin**: extension to the Kong Gateway, written in Lua or Go
- Port 8001 is used for managing APIs
- Port 8000 for service invocation



Konga – managing Kong API Gateway easier

localhost:1337#!/dashboard

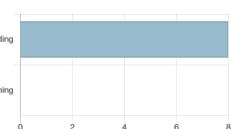
Total Requests: 71

ACTIVE	READING	WRITING	WAITING	ACCEPTED	HANDED
2	0	1	1	72	72

NODE INFO

- HostName: ip-172-31-84-175.ec2.internal
- Tag Line: Welcome to kong
- Version: 2.0.0rc2
- LUA Version: LuaJIT 2.1.0-beta3
- Admin listen: ["0.0.0.8001 reuseport backlog=16384","127.0.0.1:8444 http2 ssl reuseport backlog=16384"]

TIMERS



DATASTORE INFO Reachable

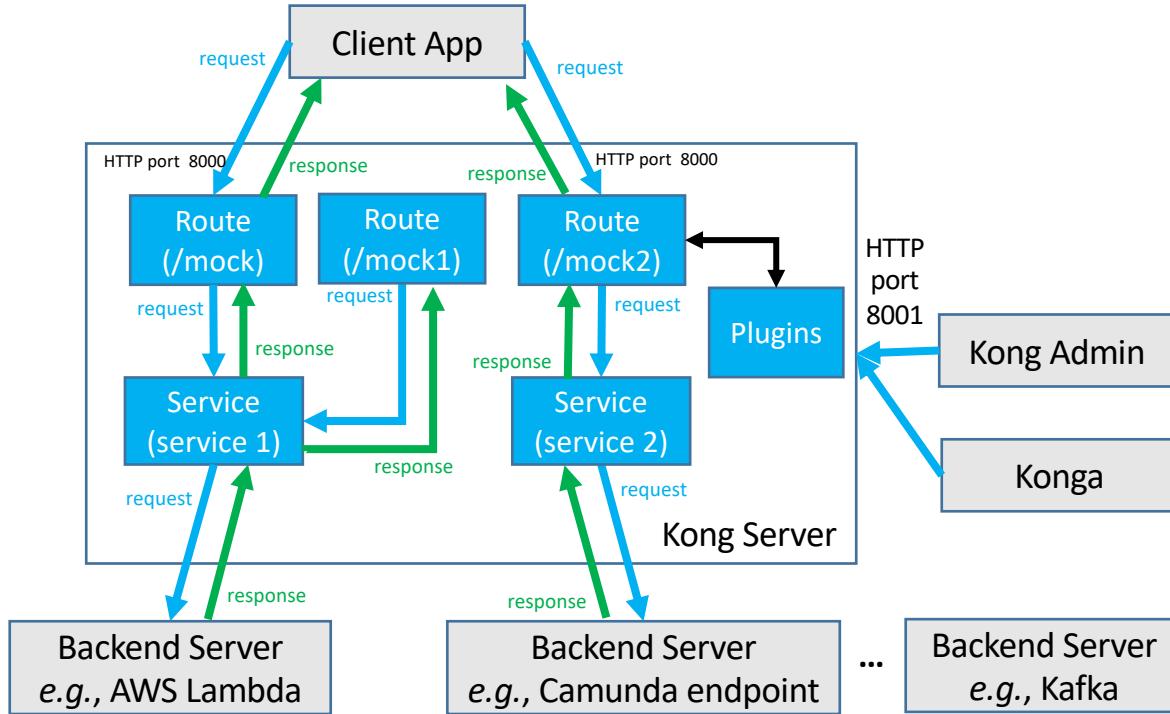
- DBMS: postgres
- Host: 127.0.0.1
- Database: kong
- User: kong
- Port: 5432

PLUGINS

- correlation-id
- pre-function
- cons
- ldap-auth
- loggly
- hmac-auth
- zipkin
- request-size-limiting
- azure-functions
- request-transformer
- oauth2
- response-transformer
- ip-restriction
- statusd
- jwt
- proxy-cache
- basic-auth
- key-auth
- http-log
- datadog
- tcp-log
- post-function
- prometheus
- adc
- syslog
- file-log
- session
- udp-log
- response-rate-limiting
- aws-lambda
- bot-detection
- rate-limiting
- request-termination

KONGA 0.14.7 GitHub Issues Support the project Connected to AWS

Konga integrated with Kong



The full lifecycle of API management

- **Design** APIs – solutions such as [swagger](#) that are able to facilitate the design, build and documentation
- **Run** APIs – solutions such as [Kong](#) that are able to connect to endpoints and return the responses
- **Secure** APIs – solutions such as [Kong plug-ins](#) that are able to apply fine-grained security and traffic policies to services
- **Govern** APIs – solutions such as [Konga](#) that are able to gain real-time visibility of the services

Enterprise Integration

Identity Management

Prof. Sérgio Guerreiro

Sergio.guerreiro@tecnico.ulisboa.pt

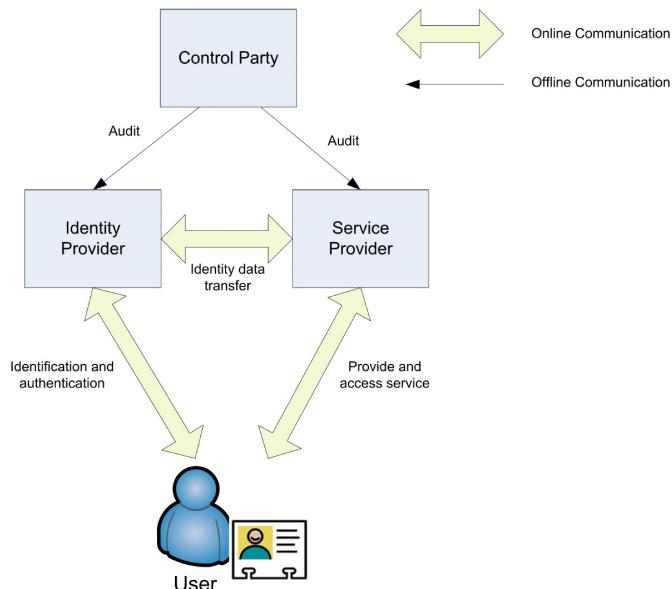
Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal

AAA framework

- **Authentication** (A) - It is the challenge process by which identity claims are tested and validated. The purpose of the authentication is ensuring that a particular user is really who he or she claims to be. *Then access is granted to a system if granted by the authorization' process execution*
- **Authorization** (A) - After authentication, the authorization process enforces the system' policies, granular access control, and user privileges. It also establishes the tasks and activities that users can perform within those authorized resources
- **Accountability** (A) - is about measuring what's happening within the system, collecting and logging data on user sessions, e.g., length of time, type of session, and resource usage. It offers a clear audit trail for compliance and business purposes

Authentication (A)



...is the **challenge** process by which identity claims are tested and validated, where challenge is secret and could be:

- Password
- Physical identifies: card, physical key
- Biometric data: face, iris, fingerprint
- Cypher/decipher challenge with keys

Authentication in applications/data stores within an organization:

- Typically, single sign-on using LDAP to interact with Active Directory, Kerberos

Figure 1: Entities involved in an identity management-system.

Authentication (A)

But, on the Cloud, multiple options possible:

- Each organization does its authentication => multiple identities
- One organization authenticates all participants
- For each business partnership, an organization (s) is chosen to authenticate
- Allow each organization to use its authentication framework and create a loosely coupled mechanism allowing the reuse of identities and authentication

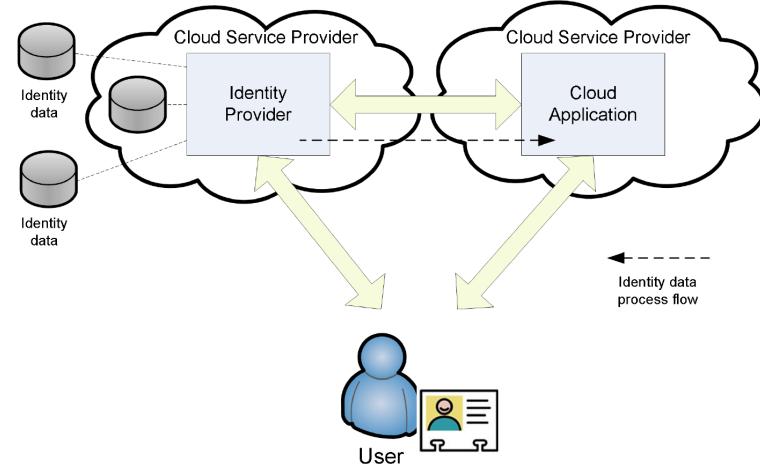


Figure 4: Identity from the Cloud-Model.

Authorization (A)

- Today, countless access control models (ACM) solutions are available in the academy and industry. Some examples:

- Discretionary access control (DAC)
- Mandatory access control (MAC)
- Role-based access control (RBAC)
- Time-role-based access control (TRBAC)
- Attribute-based access control (ABAC)
- Orcon or Chinese wall

- Nevertheless, in the majority of situations, the recognized development of ACM, these solutions specifies and implements the structural security access concerns of a single organizational silo

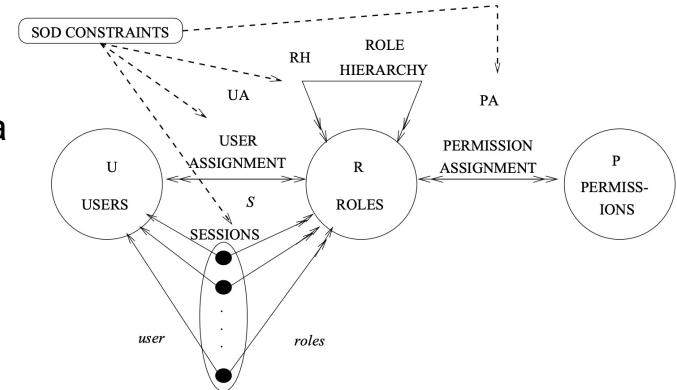


Figure 10: Symmetric RBAC—Dynamic SOD

Ravi Sandhu, David Ferraiolo, and Richard Kuhn. 2000. The NIST model for role-based access control: towards a unified standard. In Proceedings of the fifth ACM workshop on Role-based access control (RBAC '00). Association for Computing Machinery, New York, NY, USA, 47–63. <https://doi.org/10.1145/344287.344301>

eXtensible Access Control Markup Language (XACML)

- XACML is an XML-based standard markup language for specifying access control policies. The standard, published by OASIS: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml defines a declarative fine-grained, attribute-based access control policy language and a processing model describing how to evaluate access requests according to the rules defined in policies
- XACML is an attribute-based access control system
 - Input, e.g., the information about the subject accessing a resource, the resource to be addressed, and the environment
 - Output, the decision of whether access is granted or not

eXtensible Access Control Markup Language (XACML)

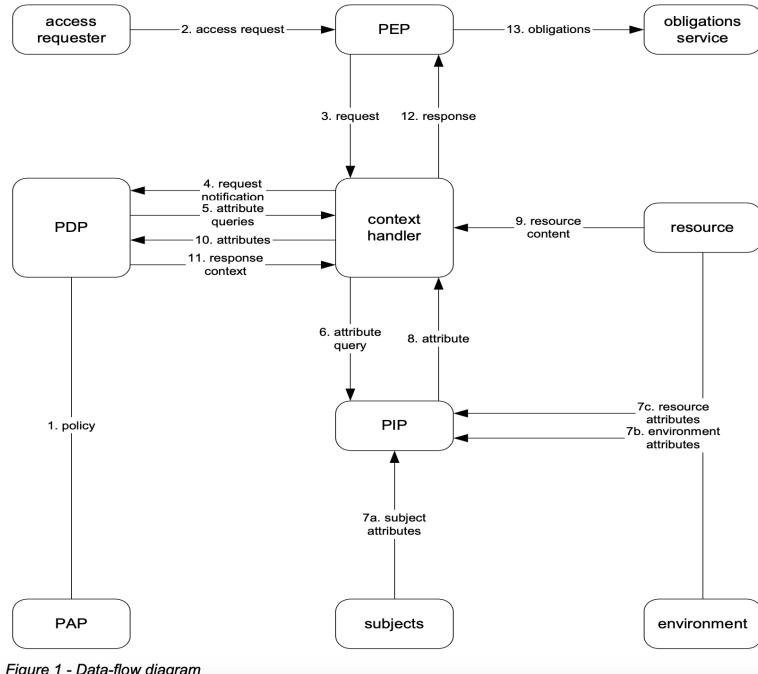
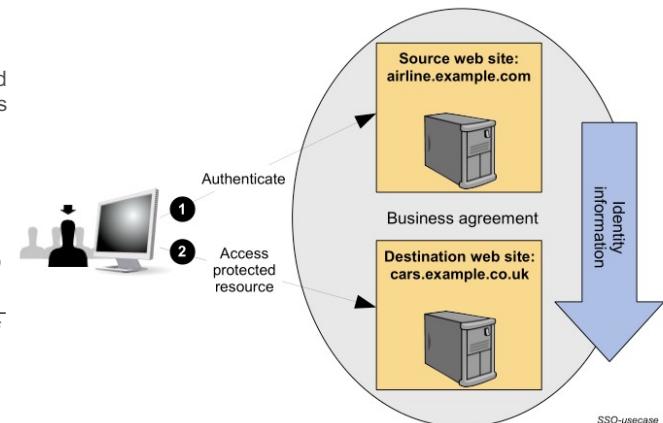


Figure 1 - Data-flow diagram

- **Policy** - A set of rules, an identifier for the rule-combining algorithm and (optionally) a set of obligations or advice. May be a component of a policy set
- **PAP** - **Policy Administration Point** is the system entity that creates a policy or policy set
- **PEP** - **Policy Enforcement Point** is the system entity that performs access control, by making decision requests and enforcing authorization decisions.
- **Obligation** - An operation specified in a rule, policy or policy set that should be performed by the PEP in conjunction with the enforcement of an authorization decision
- **PDP** - **Policy Decision Point** is the system entity that evaluates applicable policy and renders an authorization decision
- **PIP** - **Policy Information Point** is the system entity that acts as a source of attribute values

SAML

- SAML is a well-established standard developed by the OASIS consortium for authentication and authorization. Developed during early 2000s, now in the version 2.0, it is an XML-based standard heavily influenced by the enterprise technologies popular at the time, especially the WS- stack (which is a set of protocols standardized by the W3C that define how XML- and more precisely SOAP-based¹ web services should be implemented and includes publications defining SOAP, WS-Addressing, WS-Policy, WSDL etc.). Similar to some other XML-based standards, it defines a hierarchy of basic concepts - assertions, protocols, bindings and profiles. It was developed mostly for enterprise and business needs and is widely adopted in these domains as well as in public services mainly through Shibboleth, an open-source identity provider that builds upon SAML 2.0.
- The most common SAML profile (simplified, profiles are essentially use-cases) is multi-domain web single sign-on and includes interaction between three parties - a **user**, an **Identity provider**, and a **Service Provider**
- In this scenario, the user already has an established session with one web site, during which the user is redirected to a new web site. The original website acts as an IdP and relays information to the redirected website that acts as a Service provider.
- The relayed information effectively authenticates the user through a SAML message allowing for a session between the user and the redirected website to be established. In SAML terms, the first website is generally referred to as the asserting party and the second as the relying party.
- This use-case depends on various other interactions, including an agreement between parties what information to forward, what transport layer to use, whether to link user representation with both parties or to decouple them, whether to store user data or to rely on transient identifiers that get destroyed after the session is done, etc. SAML 2.0 provides mechanisms to address most of these questions on-the-fly. Additionally, there are many variations of this example flow (e.g. user-initiated instead of AP-initiated as in the example) as well as different flows supported by SAML 2.0, making it a very versatile and robust standard.
- However, this also adds complexity which is often viewed as a drawback of the standard prohibiting it from use in some other scenarios, particularly those where some of the security concerns may be somewhat relaxed. Added to that the XML legacy of the protocol is also often viewed as cumbersome and out-of-date in scenarios demanding more lightweight solutions due to processing power and other technical limitations, e.g. smartphone and similar applications.



OpenID

- OpenID Connect is an **authentication protocol** specified and maintained by the OpenID Foundation. The foundation's members are mostly international corporations including well-known names like Google, Microsoft, Oracle, PayPal, Verizon, RSA, VMWare, Deutsche Telekom
- The protocol is defined through a set of OpenID specifications including the core specification (Sakimura, 2014) and accompanying specifications detailing additional services (like dynamic provider discovery, dynamic registration with providers, response types, etc.)
- Unlike the previous versions of OpenID, the current one (final specification launched in 2014) is based on the previously described OAuth 2.0 protocol. **The specification extends and additionally specifies parts of the OAuth specification that have been left open by OAuth**, so OAuth-based mechanisms could be used in authentication scenarios
- For example, one of the most important additions by OpenID is the definition of the ID token - defined are both the contents and the representation of a data structure containing claims made by the authentication server about an end-user's authentication. Some of the other extensions introduced are a list of standard claims about a user, subject identifier types (**enabling an IdP to issue different subject value to each client for the same end-user with the purpose of disabling correlation of end-user information by different clients**), list of allowed encryptions, etc

OAuth 2.0

- OAuth 2.0 standard is published and maintained by IETF through a set of RFC documents, the core consisting of RFC 6749 (Hardt, 2012 (1)), RFC 6750 (Hardt, 2012 (2)) and RFC 6819 (McGloin, 2013). OAuth 2.0 is foremost an **delegation protocol**. The user provides permission to a third party (a client in OAuth terms) to access some of user's data at the identity. For example, the user allows a site like fit4life.com read-only access the user's fitness data stored at google.com (an authorization server and resource server in OAuth terms), without necessarily divulging any of the user's other confidential data.
- Being a more modern protocol that targets not only web applications and services but also handheld devices, OAuth makes a clear distinction between public and confidential clients based on their ability to authenticate securely with the authorization server.

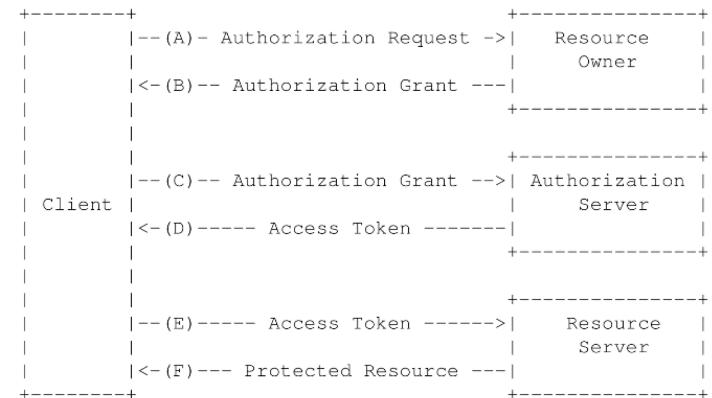
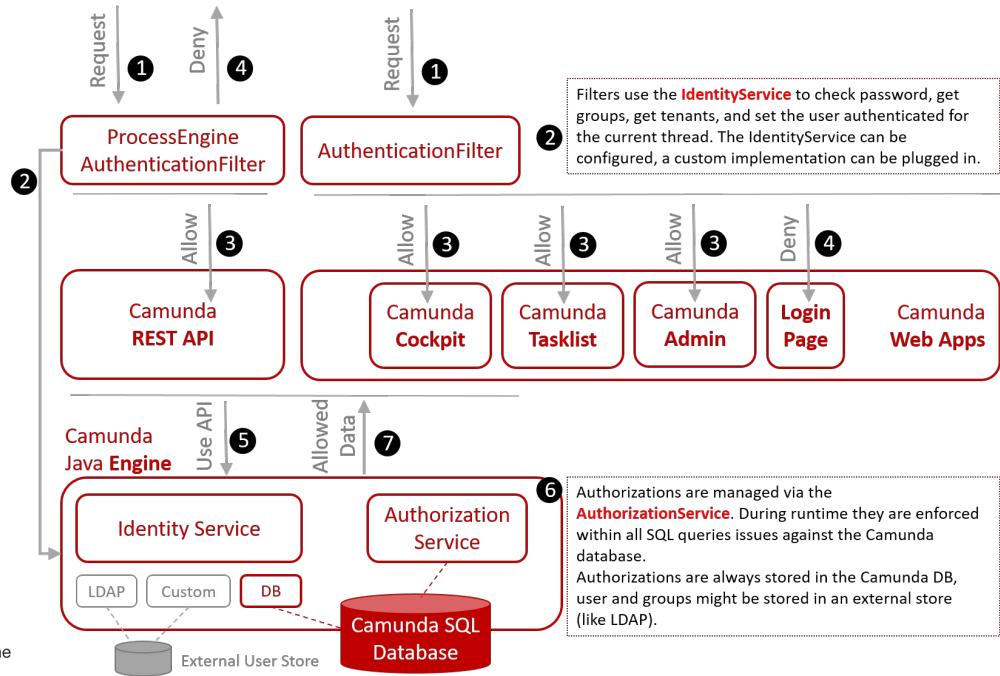


Figure 2. OAuth 2.0 Abstract protocol flow.

Camunda with Authentication and Authorization



- 1 A request is either asking for a REST API endpoint or one of the web applications functionalities.
- 2 The **ProcessEngineAuthenticationFilter** (for REST) or the **AuthenticationFilter** (for the web applications) check the user's authentication credentials via the **IdentityService**. The filters retrieve groups and tenant memberships and set the authenticated user for the current thread in the engine.
- 3 The request is *allowed*.
- 4 The request might also be *denied*, in case the authentication fails (e.g. because the username is unknown or the password does not match). For the web applications, a denied request is redirected to the login page.
- 5 All applications use Camunda's Java API internally.
- 6 Under the hood, the engine enforces authorizations by instrumenting SQL queries. That means you can never get any data from a query the current user is not authorized for.
- 7 As a consequence, only allowed and *accessible data* will be presented to the user.

Identity and Authorization services can be replaced, e.g., with OAuth2.0 or OpenID tools

OAuth2 vs. OpenID vs. SAML

	OAuth2	OpenID	SAML
Token (or assertion) format	JSON or SAML2	JSON	XML
Authorization	Delegation	No	Yes
Authentication	Pseudo-authentication	Yes	Yes
Security risks	Phishing OAuth 2.0 does not support signature, encryption, channel binding, or client verification. Instead, it relies completely on TLS for confidentiality	Phishing Identity providers have a log of OpenID logins, making a compromised account a bigger privacy breach	XML Signature Wrapping to impersonate any user
Best suited for	API authorization	Single sign-on for consumer apps	Single sign-on for enterprise Note: not well suited for mobile

Consent Management refers to the process and system by which an organization **collects, manages, and maintains user consent** for the collection and processing of personal data.

It ensures compliance with privacy laws and regulations, such as the **GDPR** (General Data Protection Regulation) and **CCPA** (California Consumer Privacy Act), by allowing users to make informed decisions about what personal data they are willing to share.

The goal is to **give users control over their data** while also providing transparency about how it is being used. It involves obtaining clear consent, documenting it, and ensuring that it can be easily modified or withdrawn by the user at any time.

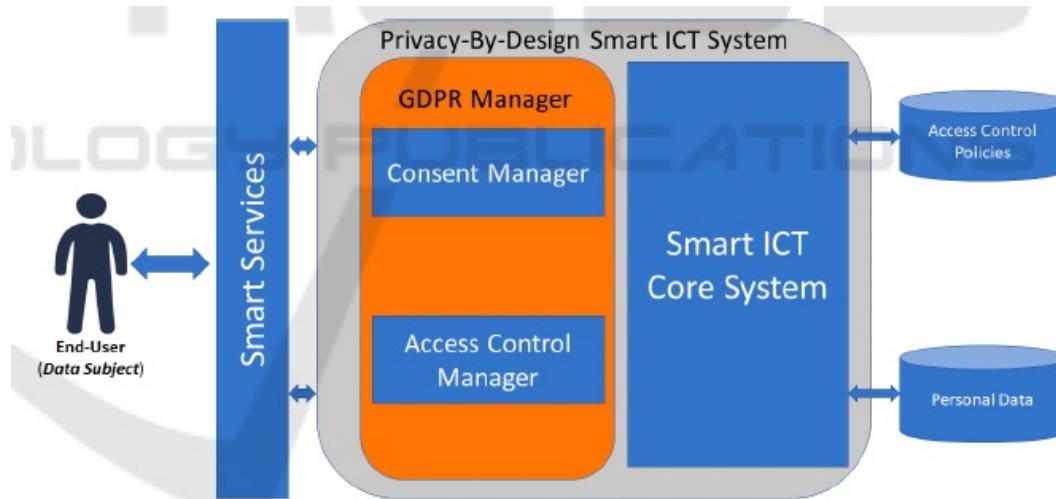
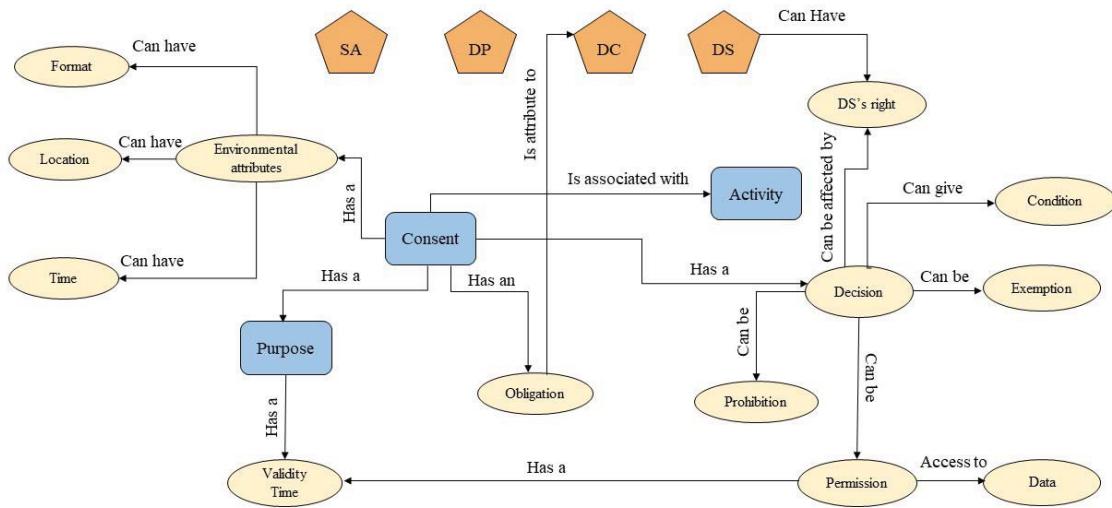
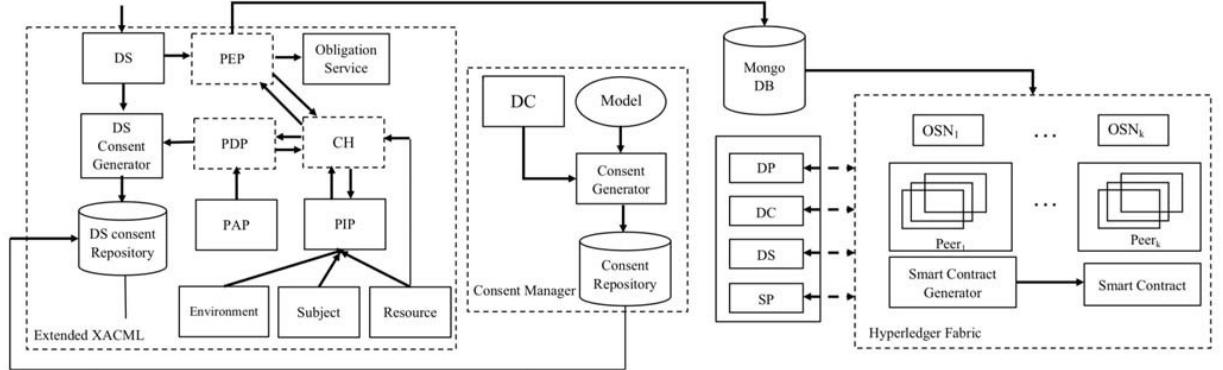
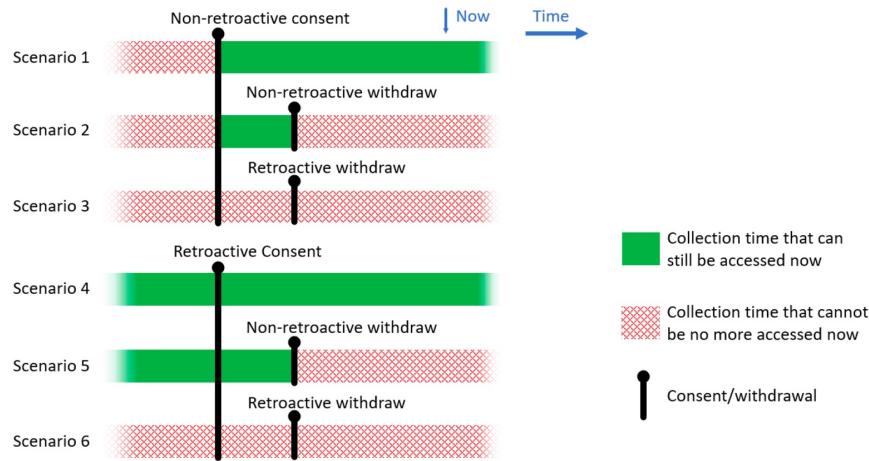
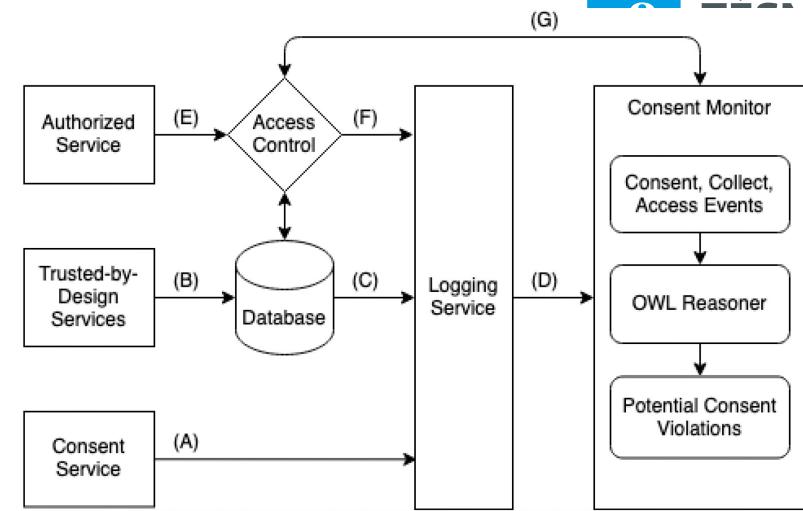
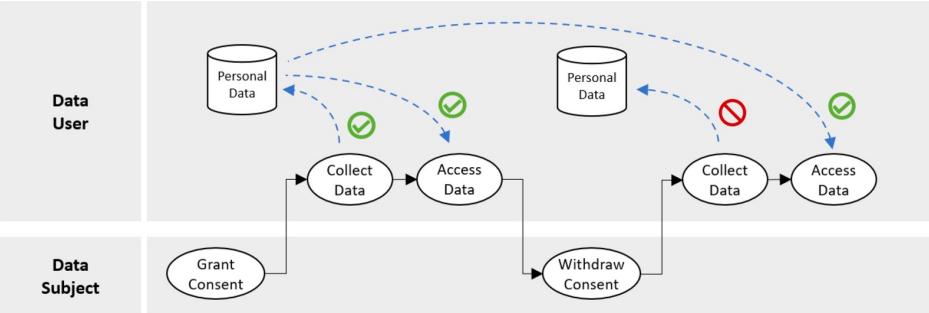


Figure 2: A Privacy-By-Design Smart ICT System Proposal.



Davari, M., & Bertino, E. (2019, December). Access control model extensions to support data privacy protection based on GDPR. In *2019 IEEE International Conference on Big Data (Big Data)* (pp. 4017-4024). IEEE.

Actor Participation


Enterprise Integration

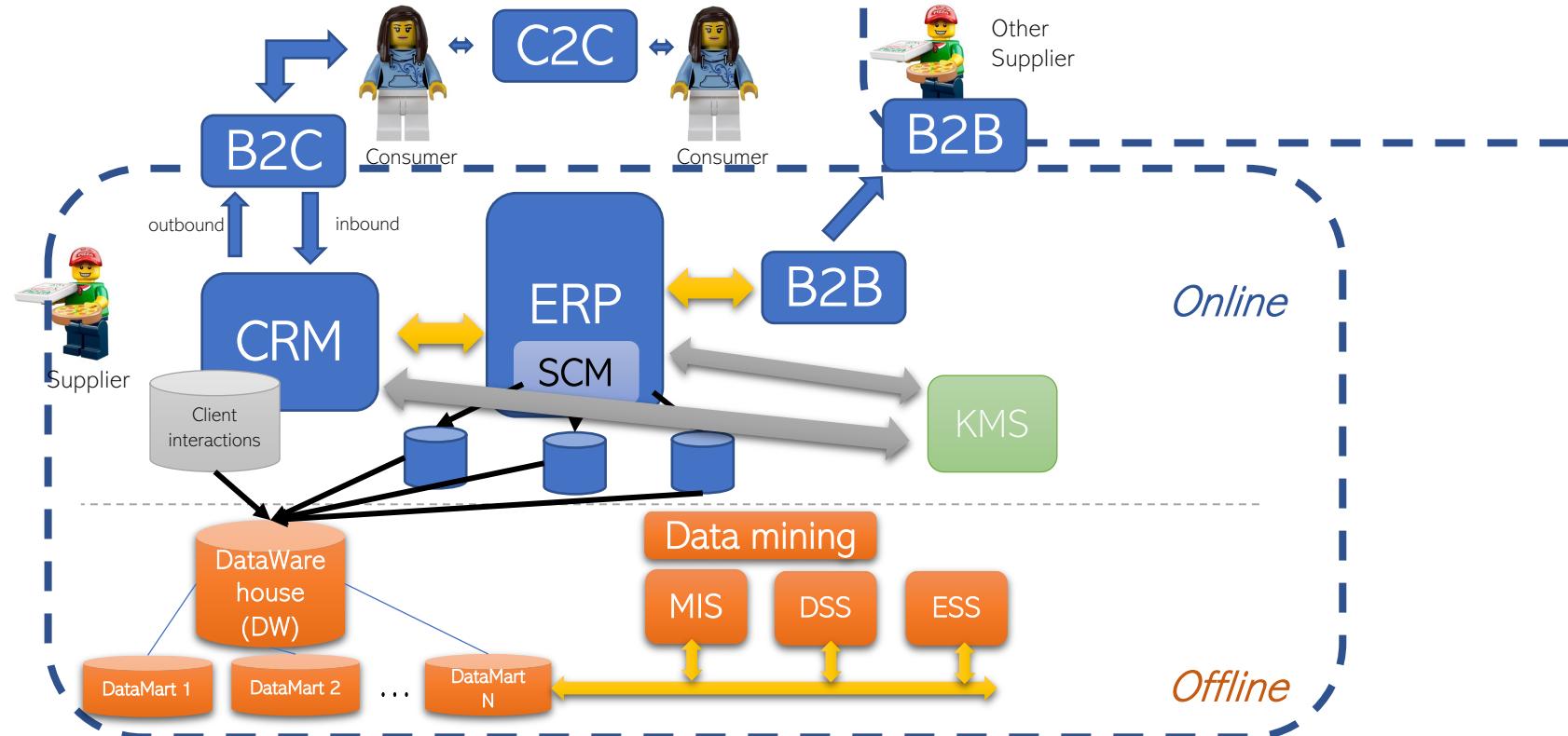
B2B

Prof. Sérgio Guerreiro

Sergio.guerreiro@tecnico.ulisboa.pt

Department of Computer Science and Engineering
Instituto Superior Técnico / Universidade de Lisboa
INESC-ID

URL: <http://www.inesc-id.pt>
Rua Alves Redol, 9
1000-029 Lisboa
Portugal



Business-to-Business (B2B) integration

SE

what is business to business?



Business-to-business (B2B) refers to the transaction of goods or services between two businesses, rather than between a business and a consumer (B2C). B2B transactions can involve the exchange of raw materials, finished products, or services that are used to support the production or operation of another business.

B2B transactions are typically characterized by larger order sizes, longer sales cycles, and more complex purchasing processes than B2C transactions. B2B businesses may also engage in partnerships and collaborations with other businesses to develop new products, expand their customer base, or improve their operations.



SE

what are the b2b integration technologies available?



There are several B2B integration technologies available that businesses can use to streamline their operations and improve collaboration with their partners. Here are some of the most common B2B integration technologies:

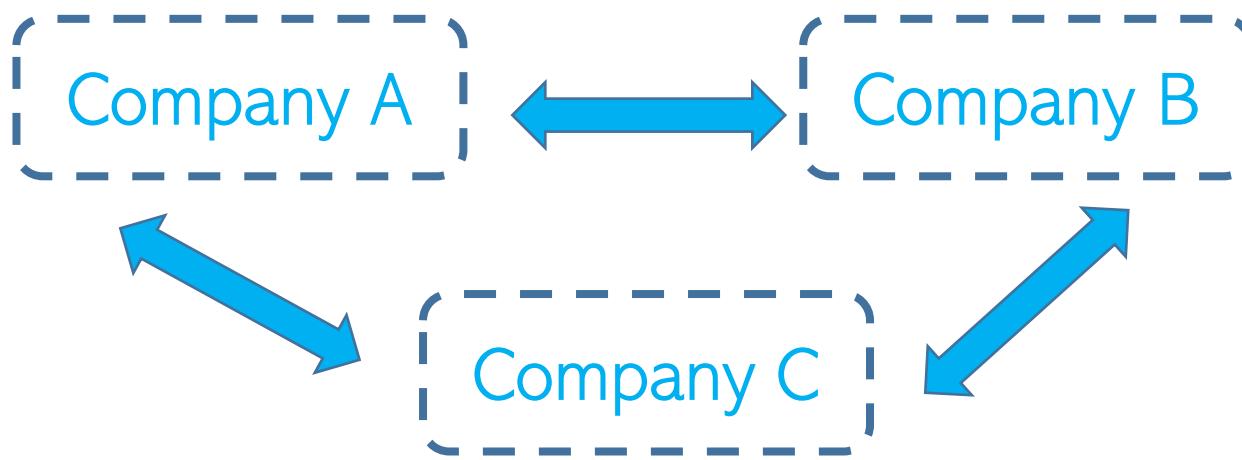
1. Electronic Data Interchange (EDI): EDI is a standard format for exchanging business documents electronically between two organizations. It allows businesses to automate the exchange of purchase orders, invoices, and other documents in a structured format, which reduces errors and improves efficiency.
2. Application Programming Interfaces (APIs): APIs are a set of protocols and tools for building software applications. B2B APIs enable businesses to integrate their systems with those of their partners, which allows them to share data and automate workflows.
3. Business Process Management (BPM): BPM is a software solution that helps businesses streamline their internal processes and improve collaboration with their partners. It provides a centralized platform for managing workflows, tasks, and approvals, which reduces errors and improves efficiency.
4. Electronic Funds Transfer (EFT): EFT is a payment technology that allows businesses to transfer funds electronically between bank accounts. It is commonly used for B2B payments, as it eliminates the need for paper checks and reduces processing time and costs.
5. Web Services: Web services are a standardized way of integrating web-based applications using open standards like XML, SOAP, and REST. B2B web services enable businesses to automate the exchange of data and transactions between their systems and those of their partners.

These B2B integration technologies can be used alone or in combination to create a customized solution that meets the specific needs of each business.



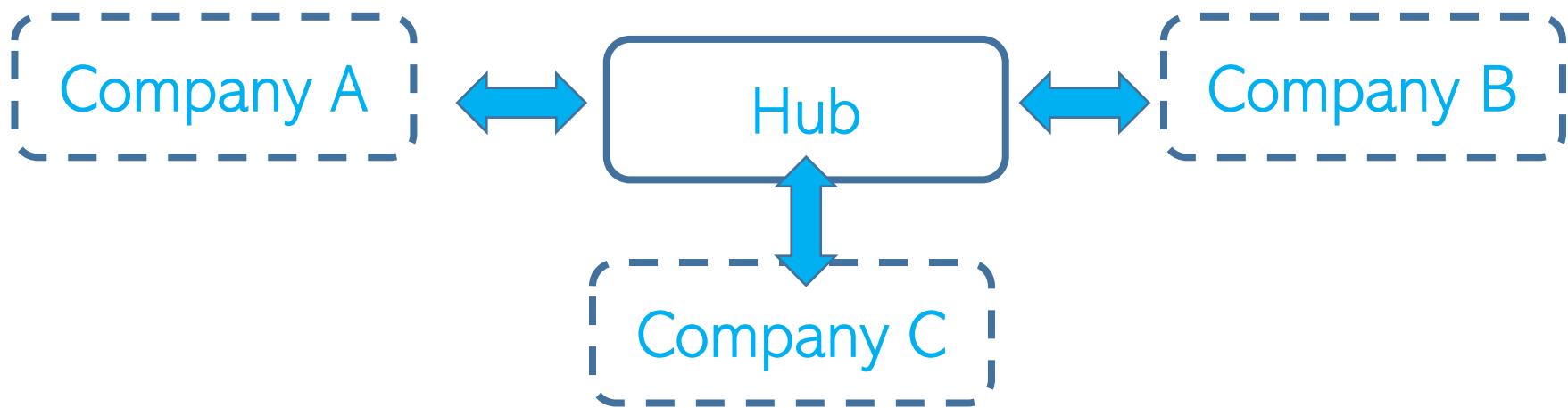
B2B – Point to point integration

- Increasing cost
- Hard to maintain and scale
 - Different protocols, messages, and security



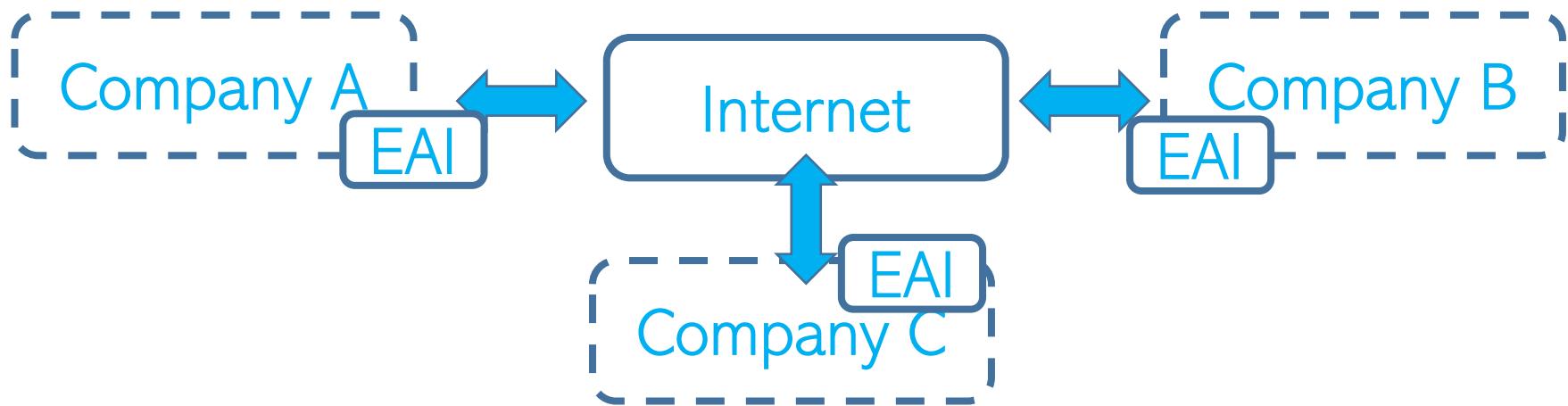
B2B – Centralized hub integration

- Who implement the bus? Who pays for it?
 - Trustability problem (*maybe solved by blockchain?*)
 - Data privacy problem
 - Cost too high, only applicable to big companies



B2B – Internet integration

- Each company is responsible for his own EAI, including all protocol layers, security and audit
- Standard EAI **connectors** simplify interoperability and reduce the deployment cost



B2B – Marketplaces

- Transaction costs is higher, but companies invest less in IS
- Trust is increased
- Value chain observability



Standards to electronic transactions between companies

- EDI
 - **1st centralized hub**; documents exchange previously agreed between partners
 - Slow and potential inconsistent formats
 - Still in use in big companies
- EDIINT – EDI over internet
- OASIS UBL – syntax recognized and understood by all, using **ebXML** specification
- Electronic invoice – VAT European directive (2004) enforced: electronic invoicing, the obligation to issue an invoice, storage of paper and electronic invoices, the details required on invoices
 - *“...VAT is applicable to the **supply of goods or services** affected for a consideration within the territory of the country by a taxable person acting as such and to the **importation of goods**...”*
- Rosetta Net (currently with GS1) – non-profit consortium aimed at establishing transaction standards, defines document and exchange protocols as part of an EDI

Q&A





TÉCNICO LISBOA