

Mainframe to Quarkus

Antoine Sabot-Durand

Chaker Fezai

Saïd Boudjelda



Antoine Sabot-Durand

- Java Champion
- Engineering director @SCIAM
- Former CDI spec lead
- Former MicroProfile Health and Fault Tolerance spec lead
- Follow me on bs: [@antoine.sabot-durand.net](https://bs.antoine.sabot-durand.net)



Saïd Boudjelda

- Tech lead @SCIAM
- Senior Backend Engineer in my daily life
- Open source contributor (May be, I can say that now)
- Some passion for FP and Distributed Systems
- Follow me: @bmscomp (GitHub)



Chaker Fezai

- Tech lead @SCIAM
- Kogito and BPMN expert
- Active tech speaker
- Experienced Java Backend Engineer
- Follow me: @fezaichaker (Twitter)



Agenda

💡 Slides here //TODO define slides url

- ⏰ Introduction context and Architecture [Antoine]
- ⏰ From EBCDIC to ASCII [Said]
- ⏰ Management of K8s jobs for the Ignite cluster. [Chaker]
- ⏰ Massive ingestion strategies [Said]
- ⏰ Kogito and BPMN for business workflow and orchestration [Chaker]
- ⏰ Other technologies used on the project [Antoine]
- ⏰ UIs for managing ingestion jobs in K8s and Kogito [Chaker]



Disclaimers



Disclaimers

- This Story is true



Disclaimers

- This Story is true
- Some names have been changed by customer request



Disclaimers

- This Story is true
- Some names have been changed by customer request
- Some facts have been simplified for the sake of clarity



Disclaimers

- This Story is true
- Some names have been changed by customer request
- Some facts have been simplified for the sake of clarity
- We won't talk about accounting business logic



Disclaimers

- This Story is true
- Some names have been changed by customer request
- Some facts have been simplified for the sake of clarity
- We won't talk about accounting business logic
- We are not Mainframe experts



Once upon a time...



Once upon a time...

- The Blue Orange insurance group decided to move its accountancy system to from the mainframe to microservices.



Once upon a time...

- The Blue Orange insurance group decided to move its accountancy system to from the mainframe to microservices.
- We're talking about 100 subsidiaries, 1 billion accounts, and 200 million movements per day...



Once upon a time...

- The Blue Orange insurance group decided to move its accountancy system to from the mainframe to microservices.
- We're talking about 100 subsidiaries, 1 billion accounts, and 200 million movements per day...
- The goal was to be able to process all this data as quickly as possible and to be able to scale the system.



Once upon a time...

- The Blue Orange insurance group decided to move its accountancy system to from the mainframe to microservices.
- We're talking about 100 subsidiaries, 1 billion accounts, and 200 million movements per day...
- The goal was to be able to process all this data as quickly as possible and to be able to scale the system.
- This talk will give you some insight on how we did it.



WAP Application at Blue Orange



WAP Application at Blue Orange

- WAP (Worldwide Accounting Platform) is the mainframe application that handles all the accounting of the group



WAP Application at Blue Orange

- WAP (Worldwide Accounting Platform) is the mainframe application that handles all the accounting of the group
- It is a monolithic COBOL application that has been in production for nearly 40 years



WAP Application at Blue Orange

- WAP (Worldwide Accounting Platform) is the mainframe application that handles all the accounting of the group
- It is a monolithic COBOL application that has been in production for nearly 40 years
- It receives a huge amount of data from all the subsidiaries



WAP Application at Blue Orange

- WAP (Worldwide Accounting Platform) is the mainframe application that handles all the accounting of the group
- It is a monolithic COBOL application that has been in production for nearly 40 years
- It receives a huge amount of data from all the subsidiaries
- A batch process runs every night to process all the data and generate the accounting entries



Wap's UP project



Wap's UP project

- Wap's up was launched at the end of 2020



Wap's UP project

- Wap's up was launched at the end of 2020
- The goal was to rewrite the WAP application in Java and to move it to a microservices architecture

Wap's UP project

- Wap's up was launched at the end of 2020
- The goal was to rewrite the WAP application in Java and to move it to a microservices architecture
- The project is expected to be completed in 2030/2031



Wap's UP project

- Wap's up was launched at the end of 2020
- The goal was to rewrite the WAP application in Java and to move it to a microservices architecture
- The project is expected to be completed in 2030/2031
- the first phase of the project (biggest) will be in service in autumn

Wap's UP project

- Wap's up was launched at the end of 2020
- The goal was to rewrite the WAP application in Java and to move it to a microservices architecture
- The project is expected to be completed in 2030/2031
- the first phase of the project (biggest) will be in service in autumn
- The team is composed of:
 - 12 Java Devs,
 - 5 teradata devs,
 - 4 front devs,
 - 10 Business Analysts,
 - 1 tech lead, 1 PMO and 2 Managers



What's a mainframe?



What's a mainframe?

- 💡 A mainframe is a large computer that is used for large-scale computing purposes that require greater availability and security than a smaller-scale machine can offer.



What's a mainframe?

- 💡 A mainframe is a large computer that is used for large-scale computing purposes that require greater availability and security than a smaller-scale machine can offer.
- 💡 Mainframes are used primarily by large organizations for critical applications, bulk data processing, and transaction processing.



What's a mainframe?

- 💡 A mainframe is a large computer that is used for large-scale computing purposes that require greater availability and security than a smaller-scale machine can offer.
- 💡 Mainframes are used primarily by large organizations for critical applications, bulk data processing, and transaction processing.
- 💡 Main language is COBOL (Common Business Oriented Language) and OS is Z/OS



Mainframes the good parts



Mainframes the good parts

- **Reliability:** They are very resilient, with built-in redundancy and failover capabilities.



Mainframes the good parts

- **Reliability:** They are very resilient, with built-in redundancy and failover capabilities.
- **Availability:** They can run for long periods of time without downtime.



Mainframes the good parts

- **Reliability:** They are very resilient, with built-in redundancy and failover capabilities.
- **Availability:** They can run for long periods of time without downtime.
- **Security:** They have robust security features, including encryption, access controls, and auditing capabilities.



Mainframes the good parts

- **Reliability:** They are very resilient, with built-in redundancy and failover capabilities.
- **Availability:** They can run for long periods of time without downtime.
- **Security:** They have robust security features, including encryption, access controls, and auditing capabilities.
- **Scalability:** They can handle large volumes of transactions and data processing, making them suitable for high-demand applications.



Mainframes the bad parts



Mainframes the bad parts

- **Cost:** They are expensive to maintain and operate.



Mainframes the bad parts

- **Cost:** They are expensive to maintain and operate.
- **Talent:** shortage of skilled mainframe operators and developers (COBOL).



Mainframes the bad parts

- **Cost:** They are expensive to maintain and operate.
- **Talent:** shortage of skilled mainframe operators and developers (COBOL).
- **Vendor lock-in:** IBM has a monopoly on the mainframe market.



Mainframes the bad parts

- **Cost:** They are expensive to maintain and operate.
- **Talent:** shortage of skilled mainframe operators and developers (COBOL).
- **Vendor lock-in:** IBM has a monopoly on the mainframe market.
- **Lack of flexibility:** They are often seen as rigid and inflexible, making it difficult to adapt to changing business needs.



Requirement to move from mainframe

Requirement to move from mainframe

- **Courage:** rewriting something that works is a big deal!



Requirement to move from mainframe

- **Courage:** rewriting something that works is a big deal!
- **Value Creation:** The new system must be able to create value for the business.

Requirement to move from mainframe

- **Courage:** rewriting something that works is a big deal!
- **Value Creation:** The new system must be able to create value for the business.
- **A good use case:** If the system you want to migrate does only data ingestion and simple consultation, it is not worth the effort.



Strategies to move from mainframe

Strategies to move from mainframe

- **Code translation:** Translate the code from COBOL to Java (or any other language). You don't change the architecture, you just change the language, and you generate non-maintainable code.



Strategies to move from mainframe

- **Code translation:** Translate the code from COBOL to Java (or any other language). You don't change the architecture, you just change the language, and you generate non-maintainable code.
- **Monolith wrapping:** Wrap the application in a concrete dome. Features are exposed as API without touching the content. You don't change the architecture, you just change the language, and you generate non-maintainable code.

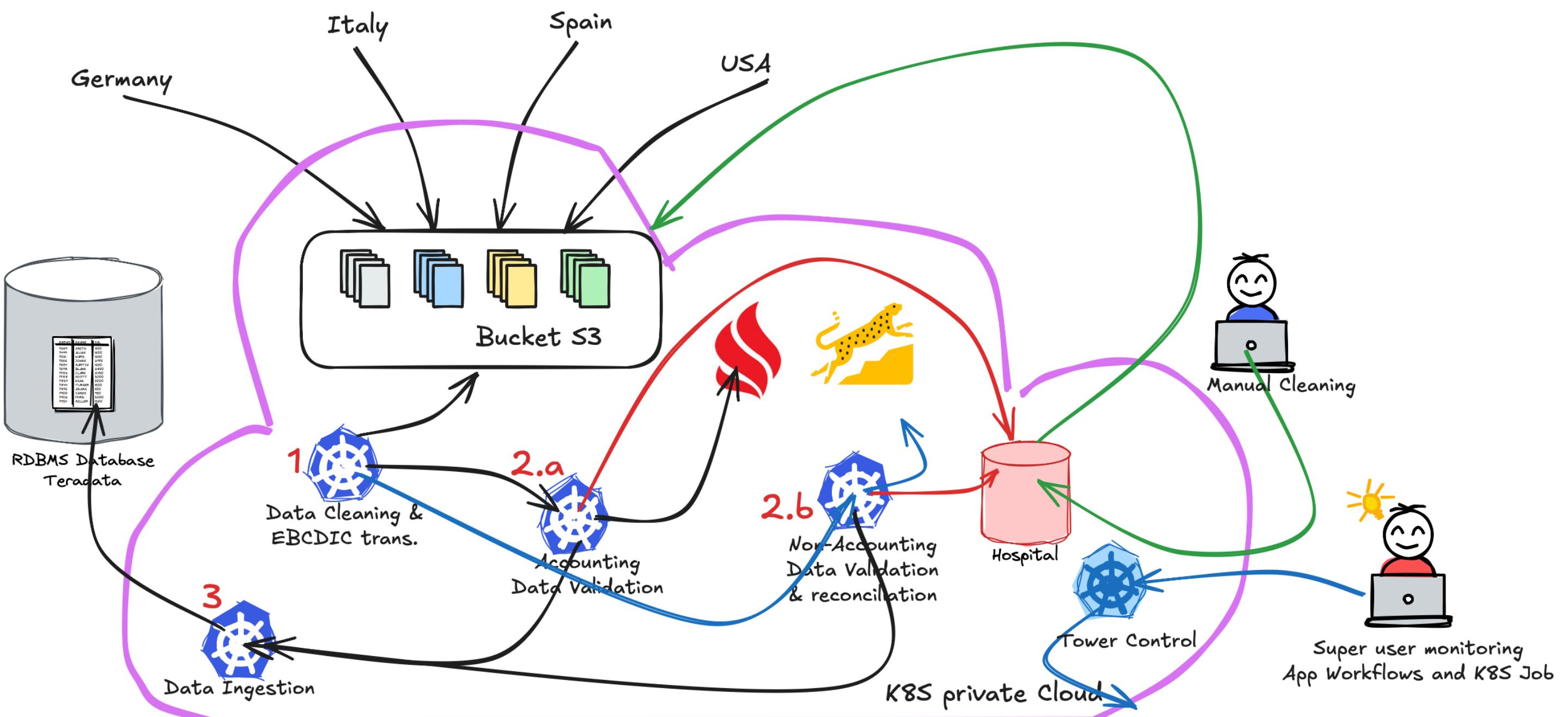


Strategies to move from mainframe

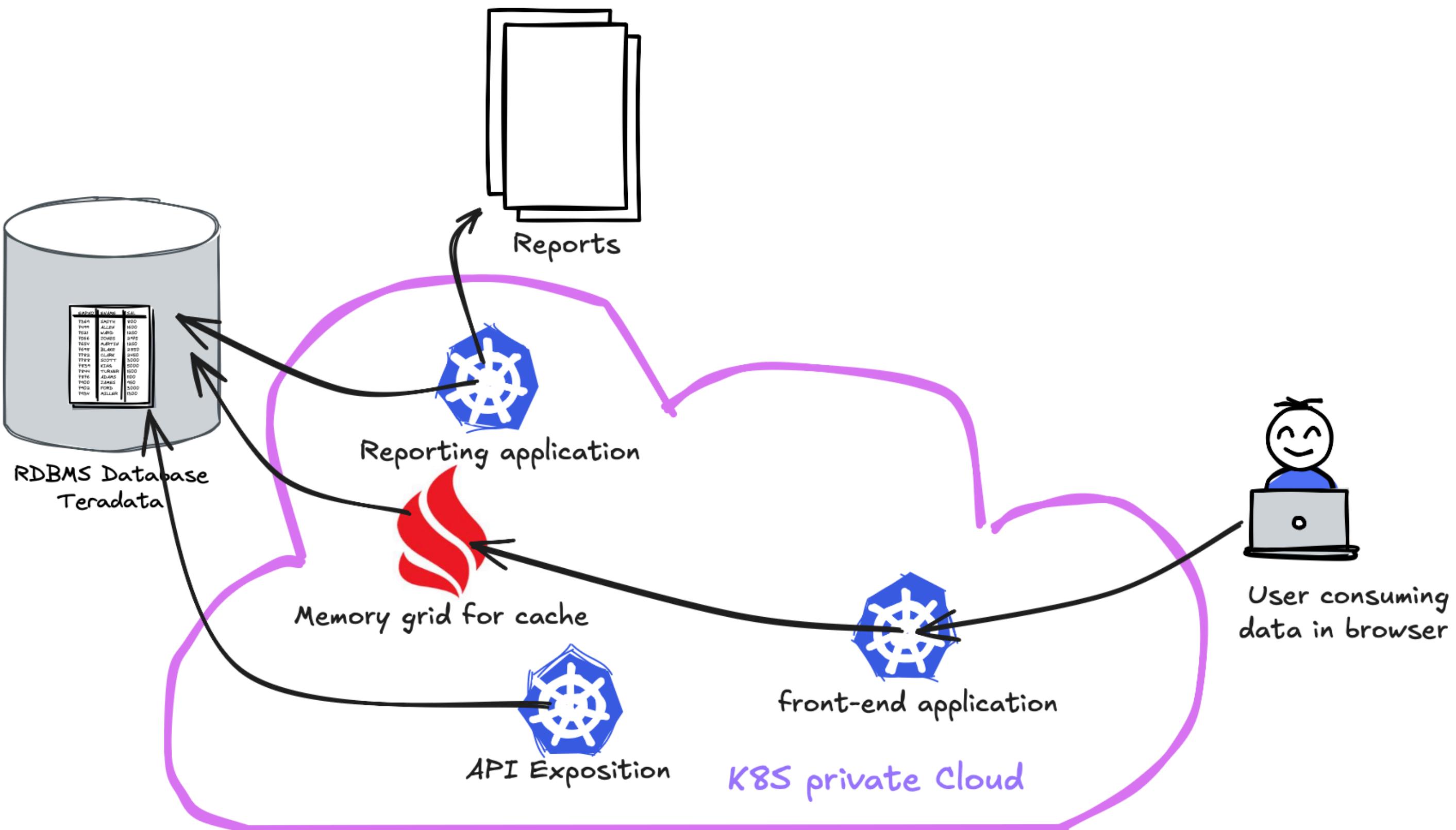
- **Code translation:** Translate the code from COBOL to Java (or any other language). You don't change the architecture, you just change the language, and you generate non-maintainable code.
- **Monolith wrapping:** Wrap the application in a concrete dome. Features are exposed as API without touching the content. You don't change the architecture, you just change the language, and you generate non-maintainable code.
- **Code rewriting:** Rewrite the code from scratch. You change the architecture and you generate maintainable code.



WAP's UP ingestion architecture



WAP's UP consuming architecture



Why Quarkus?



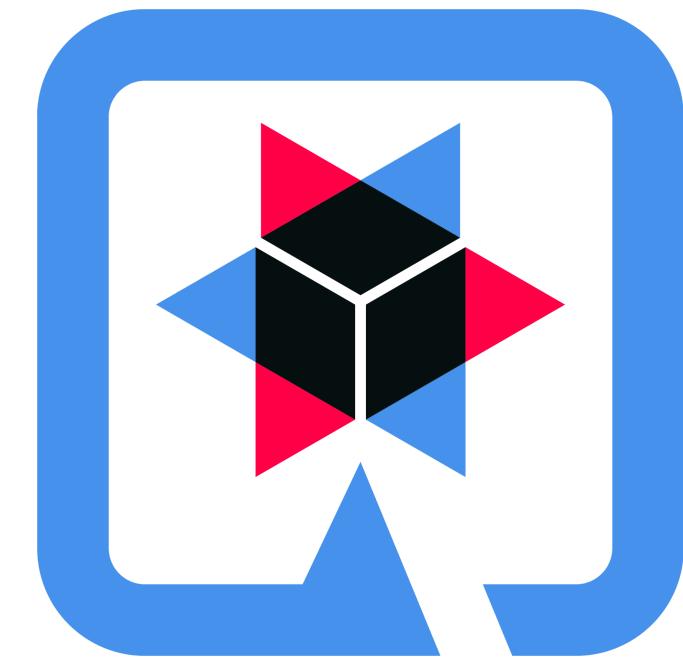
Why Quarkus?

- Quarkus is a cloud-native Java framework



Why Quarkus?

- Quarkus is a cloud-native Java framework
- It has a fast startup time and low memory footprint



Why Quarkus?

- Quarkus is a cloud-native Java framework
- It has a fast startup time and low memory footprint
- It has a huge ecosystem of libraries and extensions



Why Quarkus?

- Quarkus is a cloud-native Java framework
- It has a fast startup time and low memory footprint
- It has a huge ecosystem of libraries and extensions
- It can go further thanks to GraalVM native image



Why did we need a big SQL engine like Teradata?



Why did we need a big SQL engine like Teradata?

- Teradata is a massively parallel processing (MPP) SQL engine



Why did we need a big SQL engine like Teradata?

- Teradata is a massively parallel processing (MPP) SQL engine
- We needed to process a lot of files



Why did we need a big SQL engine like Teradata?

- Teradata is a massively parallel processing (MPP) SQL engine
- We needed to process a lot of files
- We needed a powerful SQL engine to generate huge GROUP BY and JOINs requests



EBCDIC

- EBCDIC = Extended Binary Coded Decimal Interchange Code
- Developed by IBM for **mainframes** and **midrange systems**
- 8-bit character encoding, different from ASCII
- Incompatible with ASCII – requires conversion for interoperability



Why EBCDIC Matters

- Data exchanges between modern systems and mainframes require understanding of encoding

EBCDIC character codes															
		1st hex digit													
		2nd hex digit													
0	NUL	DLE	DS		SP	&	-								0
1	SOH	DC1	SOS				/	a	j			A	J		1
2	STX	DC2	FS	SYN				b	k	s		B	K	S	2
3	ETX	TM						c	l	t		C	L	T	3
4	PF	RES	BYP	PN				d	m	u		D	M	U	4
5	HT	NL	LF	RS				e	n	v		E	N	V	5
6	LC	BS	ETB	UC				f	o	w		F	O	W	6
7	DEL	IL	ESC	EOT				g	p	x		G	P	X	7
8		CAN						h	q	y		H	Q	Y	8
9		EM						i	r	z	'	I	R	Z	9
A	SMM	CC	SM		CENT	!	:								
B	VT	CUI	CU2	CU3		\$,	#							
C	FF	IFS		DC4	<	*	%	@							
D	CR	IGS	ENQ	NAK	()	-	'							
E	SO	IRS	ACK		+	:	>	=							
F	SI	IUS	BEL	SUB		--	?	"							

Figure 1. EBCDIC Character Set

COBOL COPYBOOKS

```
01 CUSTOMER-RECORD.  
  05 CUST-ID          PIC 9(5).  
  05 CUST-NAME        PIC A(30).  
  05 CUST-ADDRESS      PIC A(50).  
  05 CUST-BALANCE     PIC S9(9)V99 COMP-3. * Ex : ±999999999.99
```

- Standard record definitions for reading EBCDIC-encoded files
- Used to define the structure of records in COBOL programs
- Ensures consistency in data structure definitions across COBOL programs
- Often used in conjunction with data conversion routines (e.g., EBCDIC to ASCII)

How to read EBCDIC files

- **CFT Axway** (We are not allowed)
- **Java™ Batch Launcher and Toolkit for z/OS® (JZOS)** (No one wants to touch mainframes)
- **Cobrix - COBOL Data Source for Apache Spark** (We have to drag the complexity of Apache Spark)



EBCDIC files, The JVM way

- Code something from scratch, (We had little time!!)
- **JRecord** (Our best fit)



JRecord a road to Contribution

- Open-source Java library available on GitHub github.com/bmTas/JRecord
- You can contribute to it, everything is welcome
 - Use it and test it
 - Report issues
 - Bug fixes
 - New features
 - Documentation

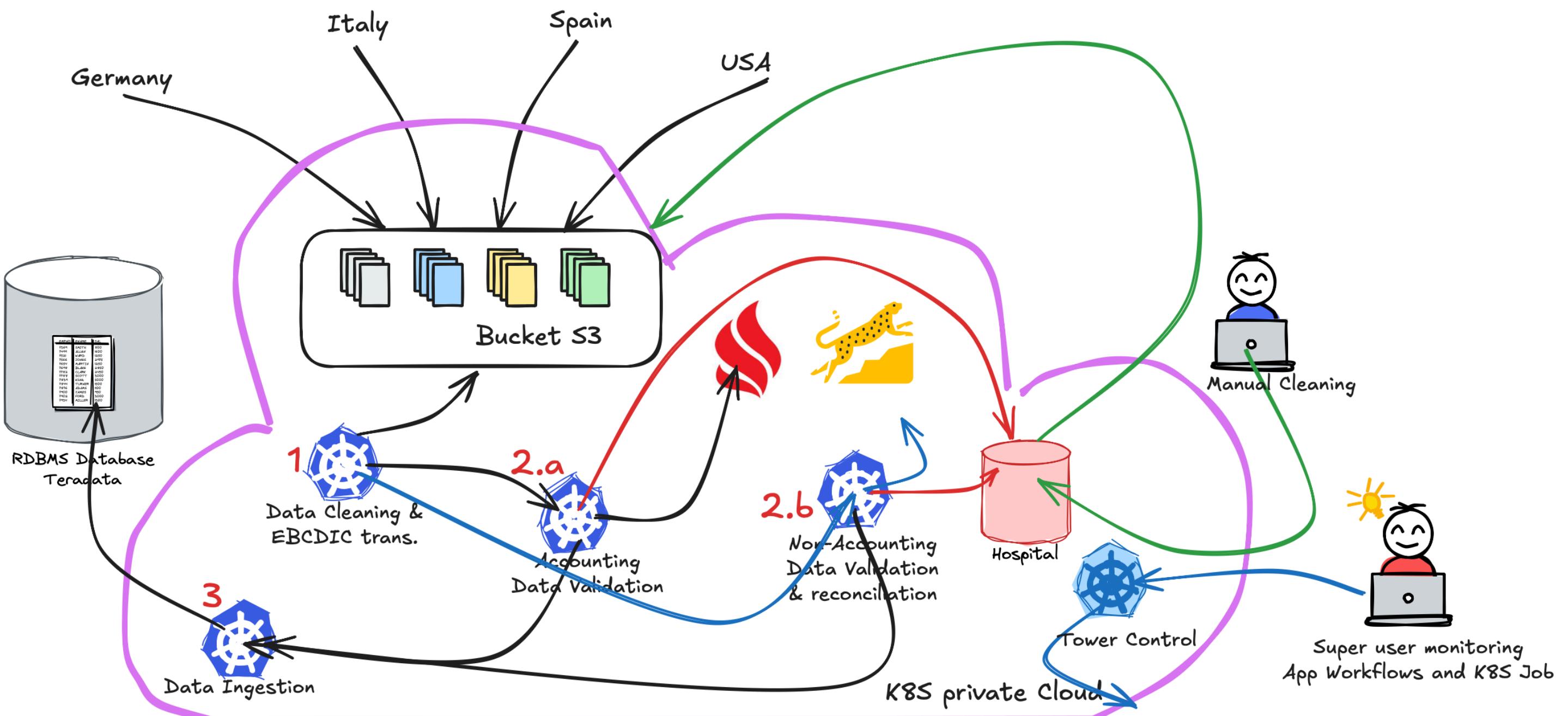


JRecord, time to play with (Very Small Demo)

- Prepare an EBCDIC file
- Prepare a Cobol Copybook
- Prepare a Java program
- Start reading EBCDIC file and write into ASCII file



High-Performance Distributed File Processing



Stack Technique

- **Kubernetes:** Orchestrates and dynamically creates pods
- **Fabric8 API:** Triggers pod creation based on the file size
- **Apache Ignite:** In-memory distributed storage for file lines
- **Java Quarkus:** Lightweight JVM application in each pod, fast startup
- **COS/S3:** Source of the files to be processed



K8S



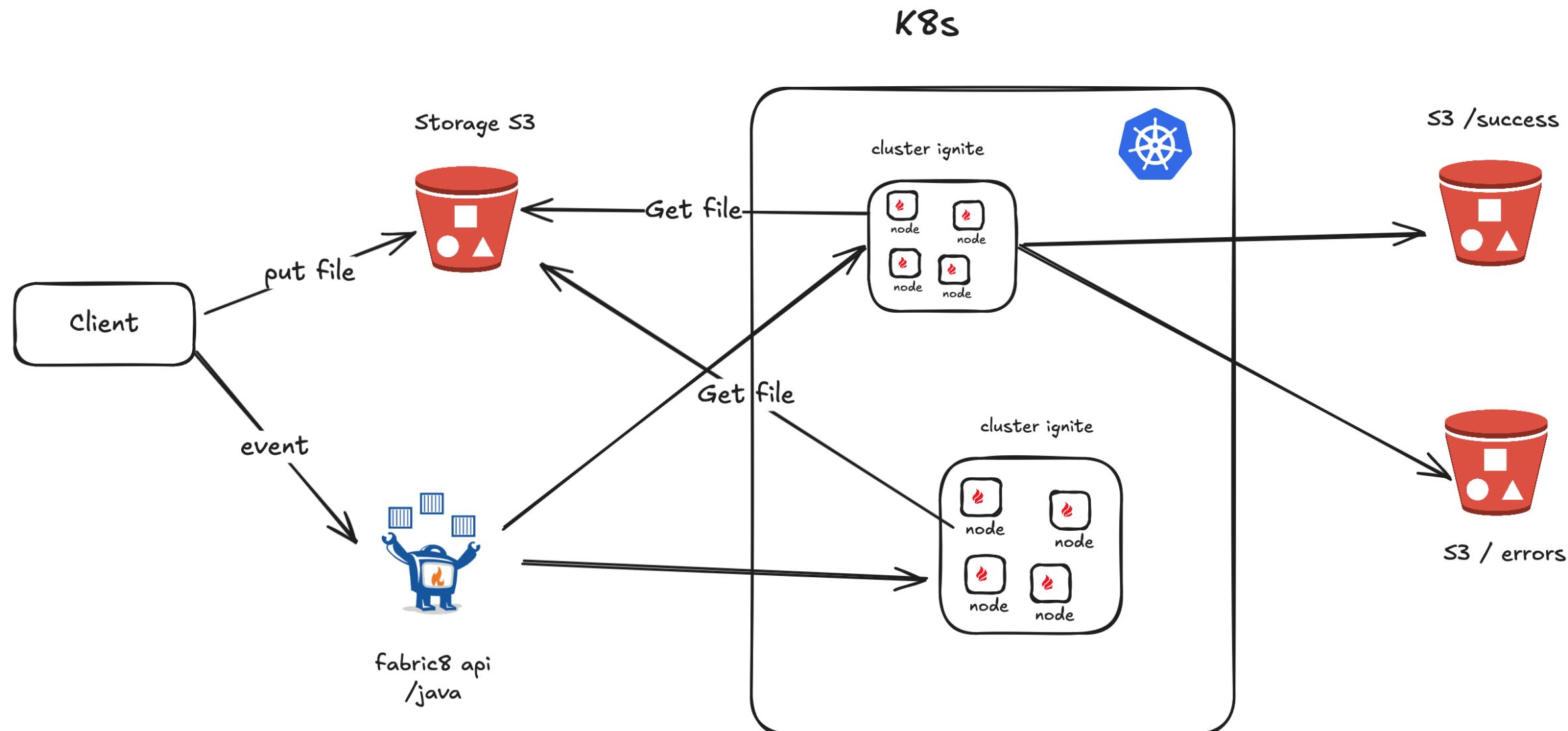
Ignite



Fabric8

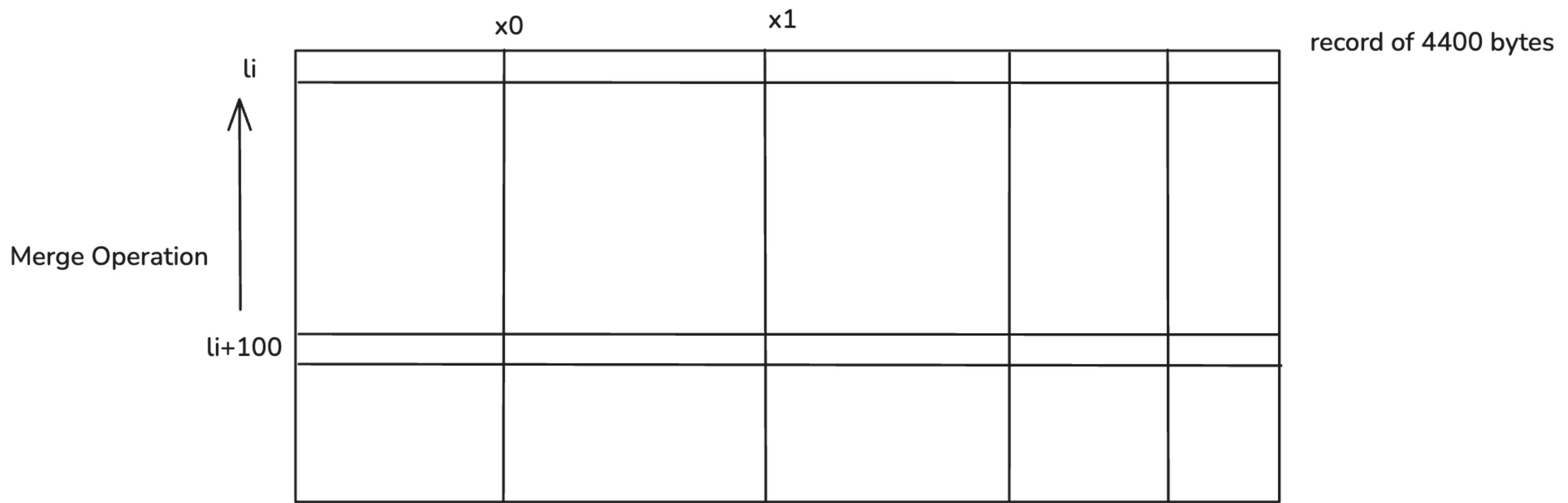


Quarkus



The ingestion challenge

- Convert EBCDIC files to ASCII (UTF-8)
- Slice the files into chunks (Skip bad lines)
- Validate each partition (Complex validation rules)



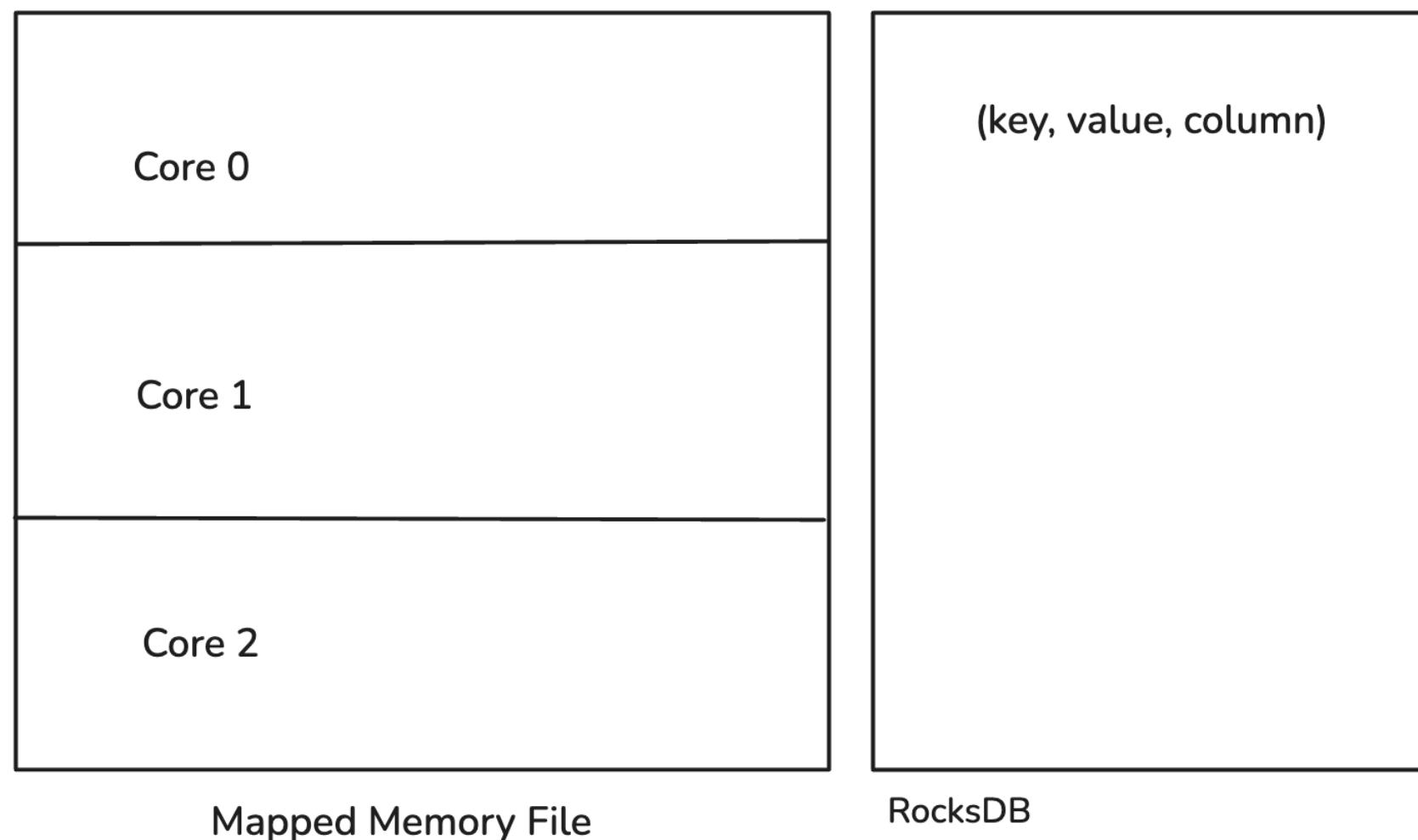
Worst ideas ever (Do not do that)

- Use old-fashioned Java IO (BufferedReader)
- Read the content of the file and store into the heap
- Collision issue (Even with a good hash function)
- OutOfMemory and heap space issues
- Garbage collector tuning headache



What will be the solution then?

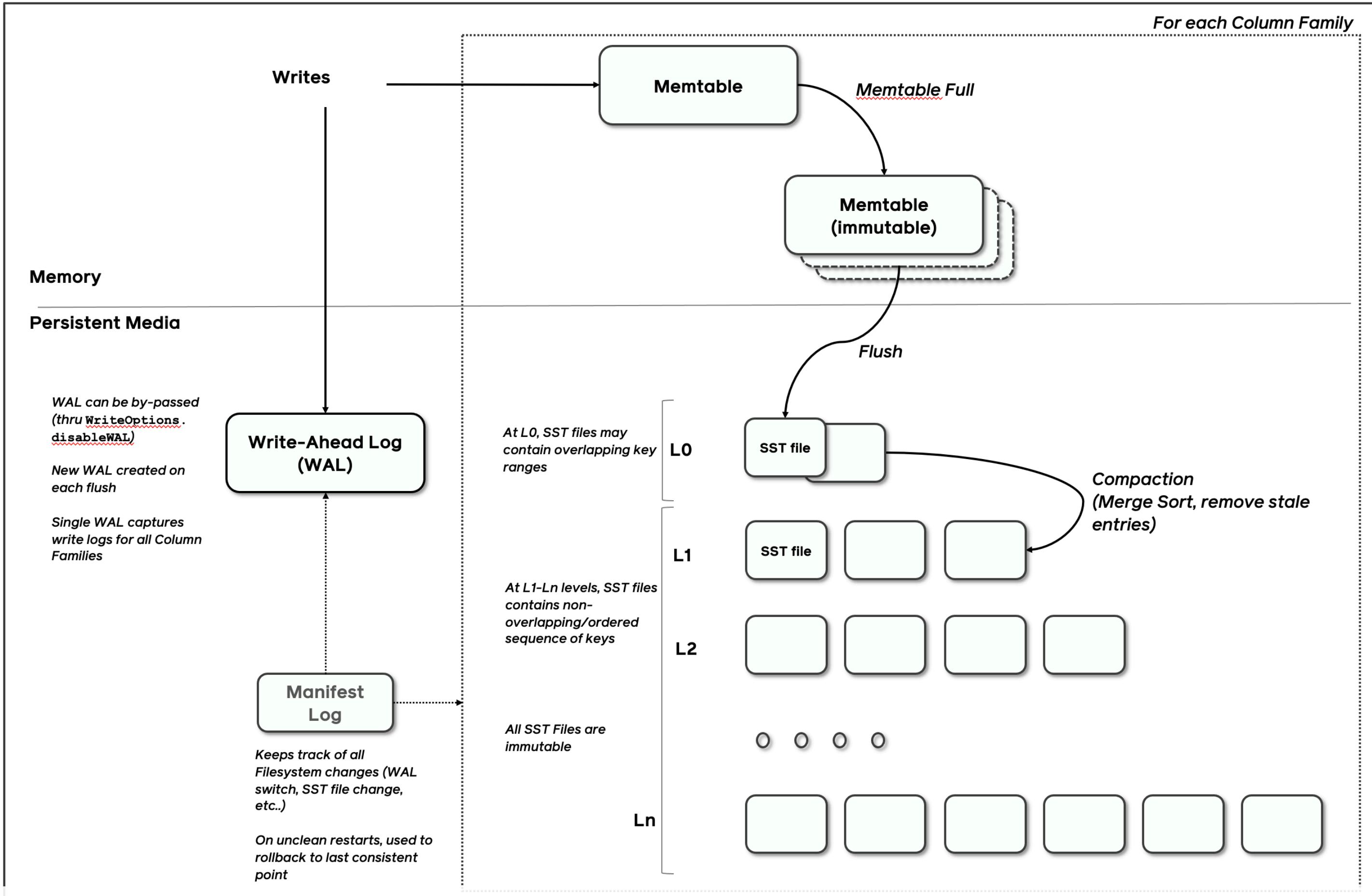
- Use Fixed Length files (more control of memory)
- Use RocksDB as a key-value store (Fast and Reliable)
- Use Memory Mapped Files (FFM API de JDK 23)
- Use multi threading for file processing (More CPU usage)



RocksDB

- RocksDB is an embeddable, persistent key-value store for fast storage
- It is a C++ library built on LevelDB
- It is optimized for fast storage (SSD and Flash)
- It is designed for fast read and write operations
- It is used by many big companies (Facebook, LinkedIn, Yahoo, etc.)

RocksDB, High Level Architecture



Memory Mapped Files vs. BufferedReader

Aspect	BufferedReader	Memory-Mapped Files
Speed	Slower (sequential)	Faster (direct OS access)
Memory Overhead	Heap (buffer-controlled)	Off-heap (OS-managed)
Access Pattern	Sequential only	Random access (like an array)

Buffered Reader

```
// Sequential, line-by-line reading
try (BufferedReader br = new BufferedReader(new FileReader("file.txt"))) {
    String line;
    while ((line = br.readLine()) != null) { /* Process line */ }
}
```

- Pros:
 - Handles text encoding automatically.
 - Safe for small-to-medium files.
- Cons:
 - Slower for large files (multiple syscalls).
 - Heap memory usage.



Memory-Mapped Files

```
// Random access via OS mapping
try (RandomAccessFile raf = new RandomAccessFile("file.bin", "r")) {
    FileChannel channel = raf.getChannel();
    MappedByteBuffer buffer = channel.map(READ_ONLY, 0, channel.size());
    while (buffer.hasRemaining()) { byte b = buffer.get(); /* Direct access */ }
}
```

- Pros:
 - Near-native speed (bypasses JVM heap).
 - Efficient for large/binary files.
- Cons:
 - Complex error handling (e.g., `OutOfMemoryError` if mapping fails).
 - OS-dependent (limited by virtual address space).



Time for a small Demo

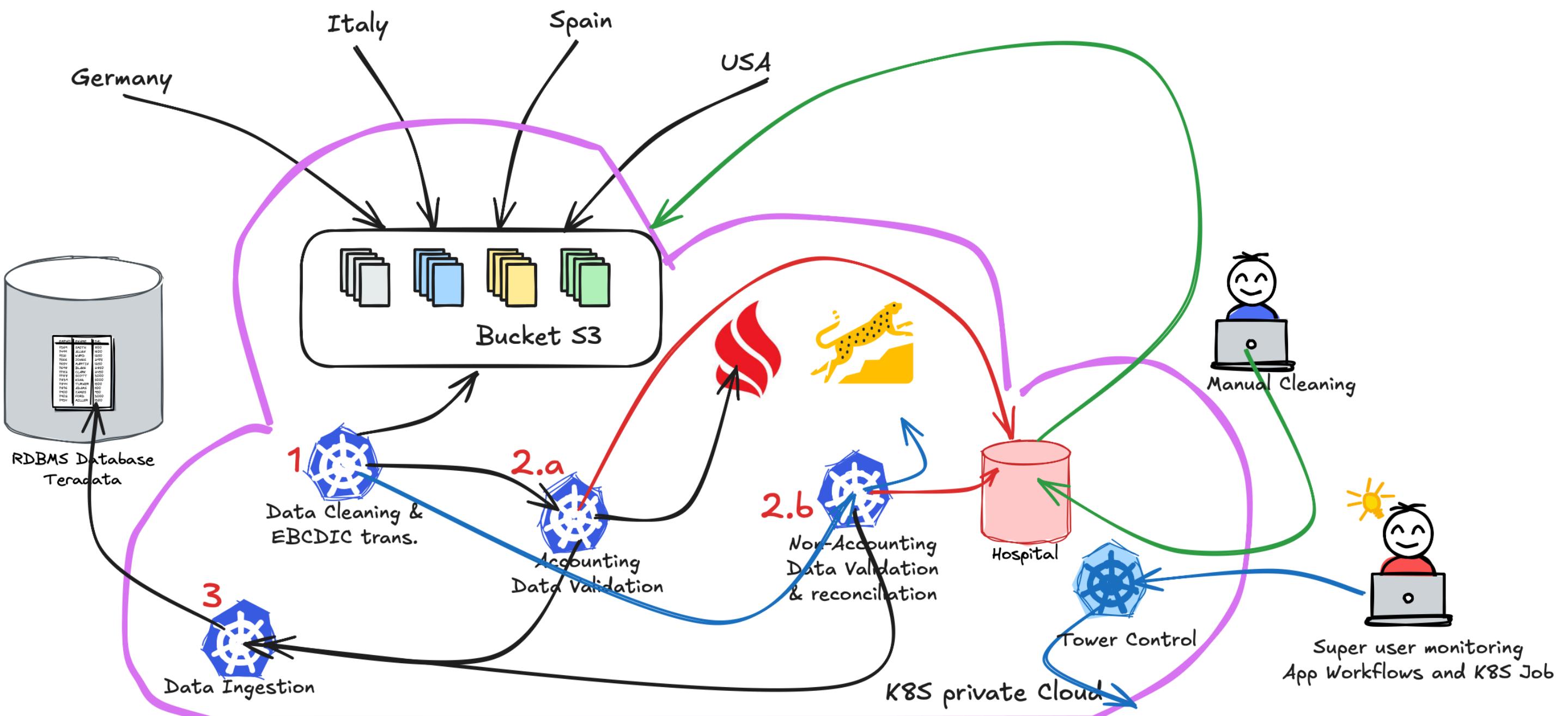
- Read using both ways and compare the results



Orchestration

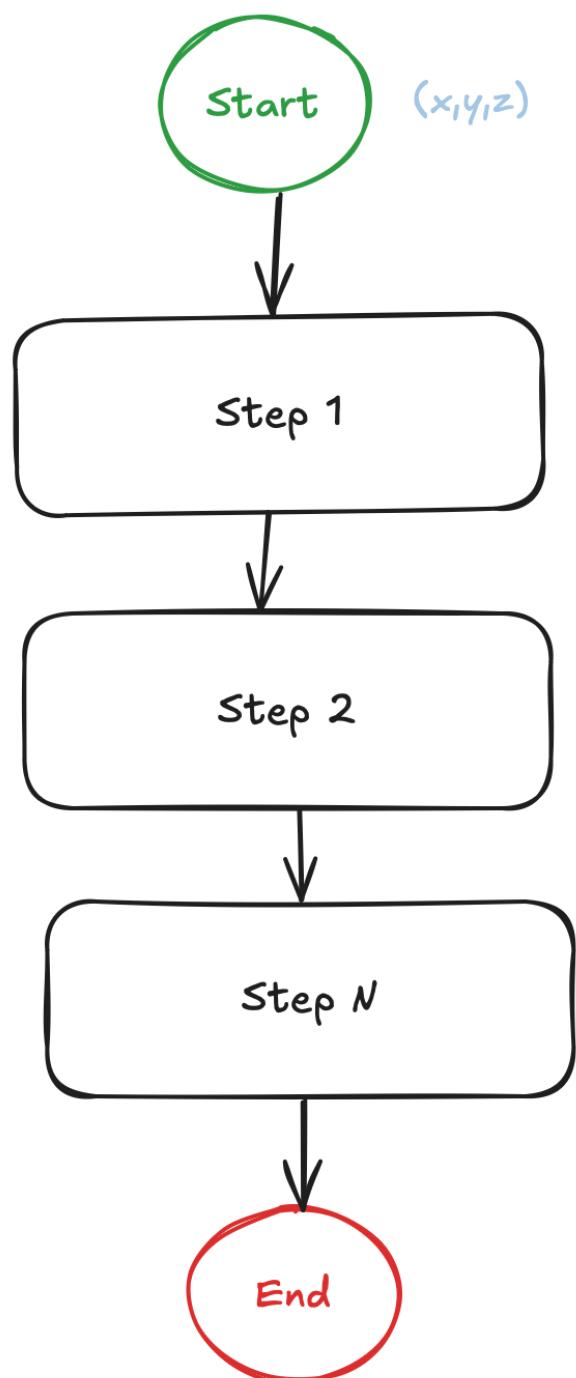


Orchestration

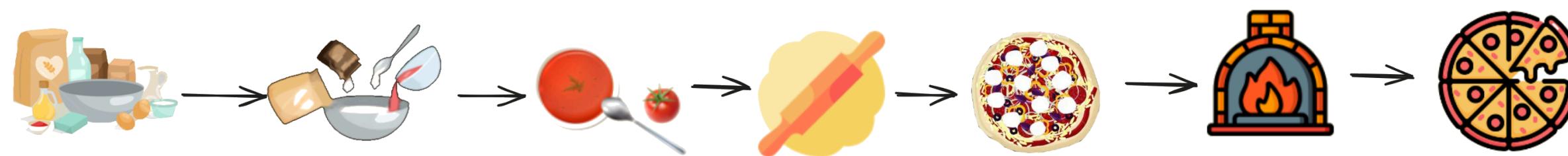


Workflow Technologies

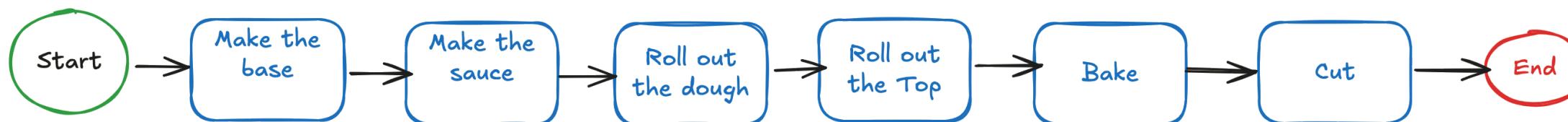
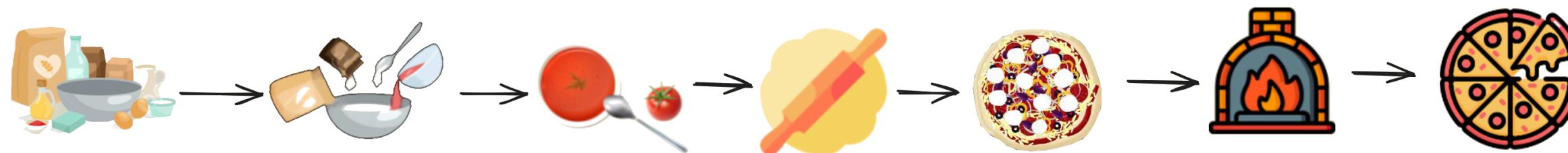




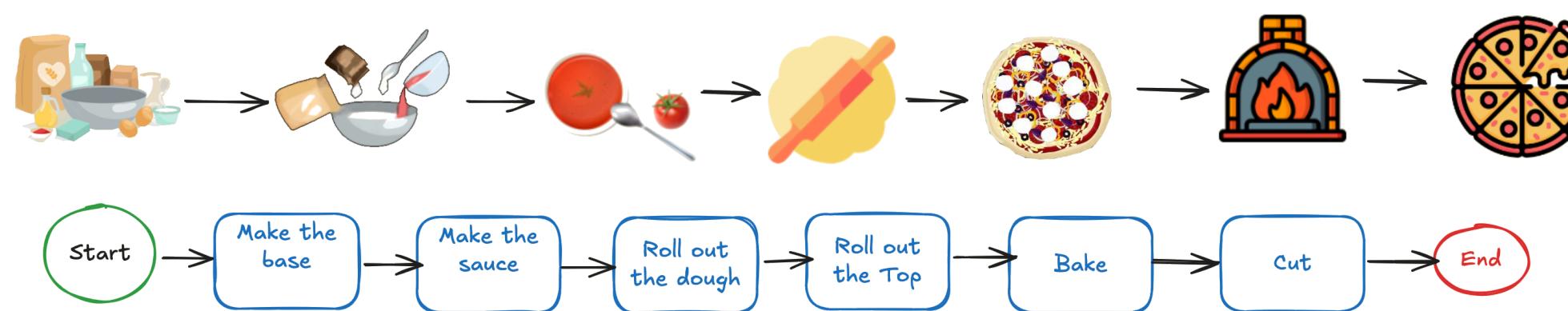
Cooking Pizza Workflow



Cooking Pizza Workflow

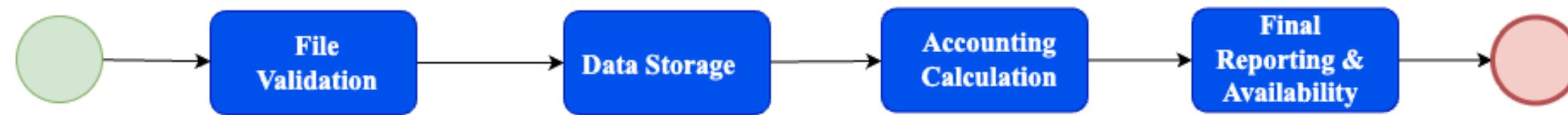


Cooking Pizza Workflow

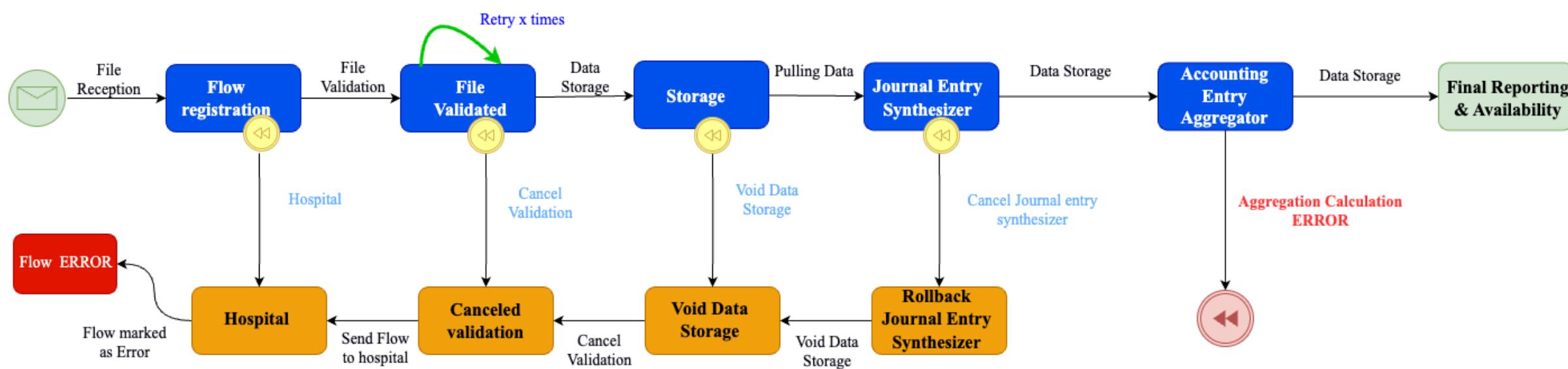


```
public class Pizzeria {  
    ...  
    public Pizza cookingPizza(int numberofPizzas, int time) {  
        //Make pizza dough  
        var dough = beaterService.makePizzaDough(numberofPizzas);  
        //Roll out the dough and top  
        var rollOutTop = rollOutService(dough);  
        //Bake the pizza  
        var pizza = ovenService.bake(rollOutTop, time);  
        // Cut pizza  
        pizza = ovenService.cut(pizza);  
        return pizza;  
    }  
}
```

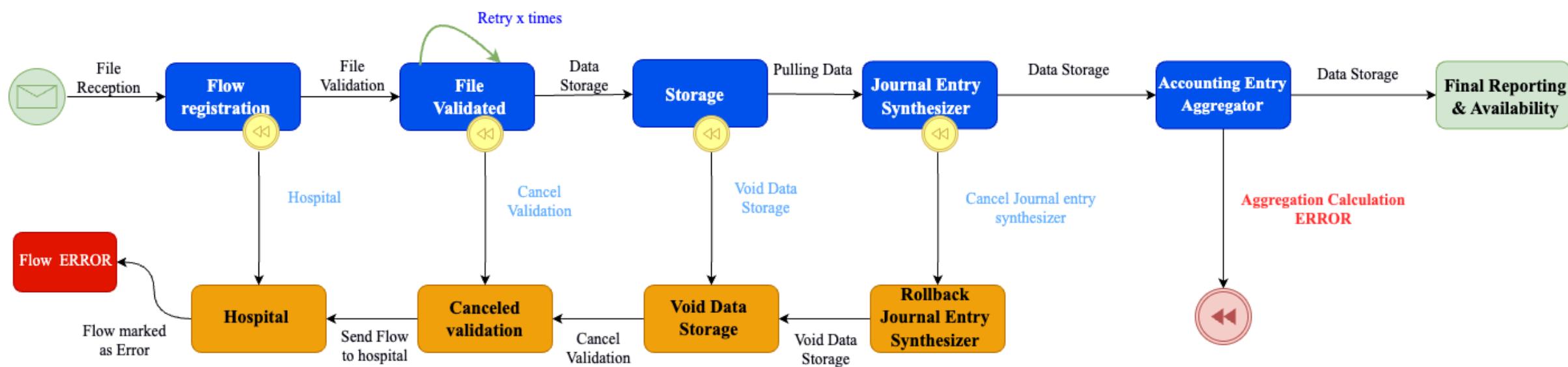
Accounting Calculation Workflow



Accounting Calculation Workflow



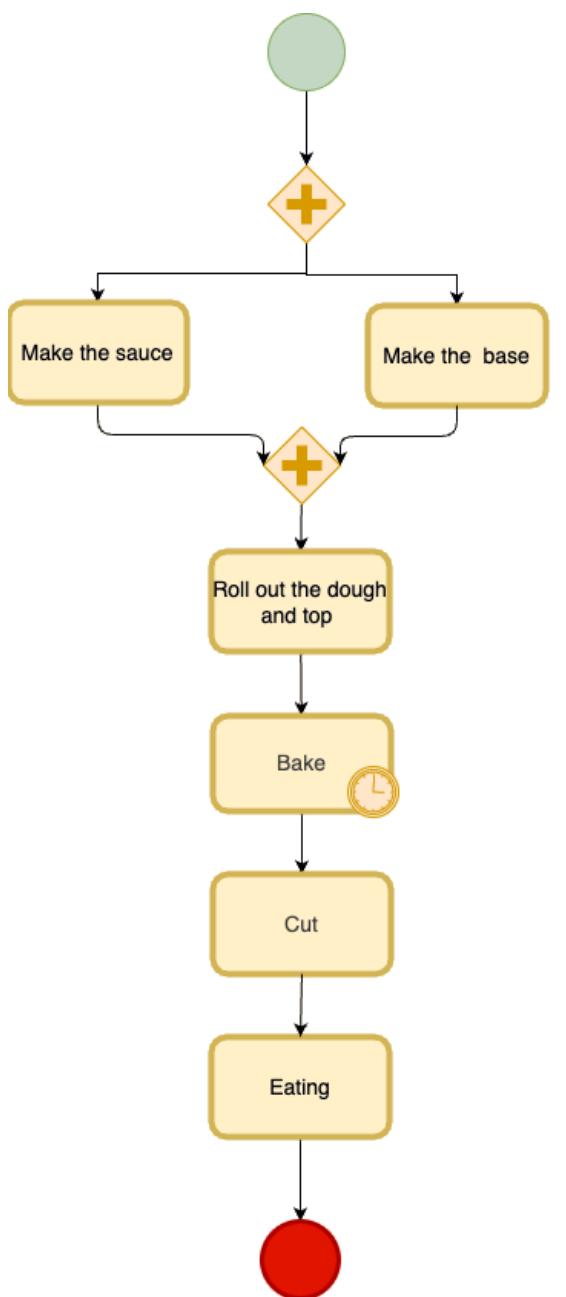
Accounting Calculation Workflow



BPMN 2.0



BPMN 2.0



Events



Tasks



Gateways



Connectores



Kogito



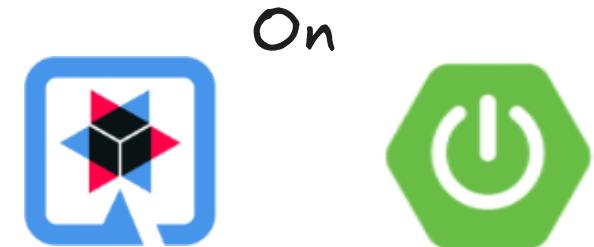
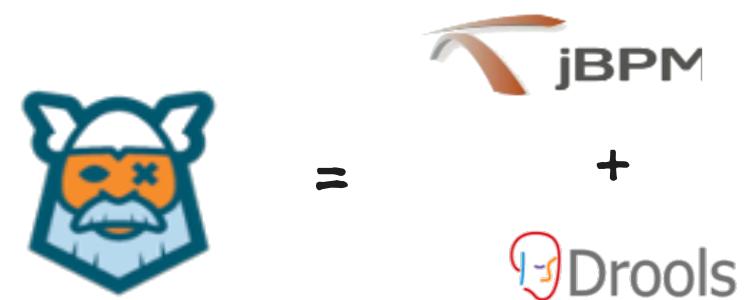
Kogito

Next-gen Cloud native Business Automation



💡 Cogito ergo sum

Cloud native Business Automation for building Intelligent applications backed by battle-tested capabilities



Traditional platform BPMN

Traditional platform BPMN

-  A large monolith and Stateful

Traditional platform BPMN

- A large monolith and Stateful
- Not Optimized for cloud and architecture microservices.

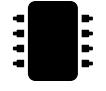
Traditional platform BPMN

-  A large monolith and Stateful
-  Not Optimized for cloud and architecture microservices.
-  Long startup time

Traditional platform BPMN

-  A large monolith and Stateful
-  Not Optimized for cloud and architecture microservices.
-  Long startup time
-  Large compute consumption

Traditional platform BPMN

-  A large monolith and Stateful
-  Not Optimized for cloud and architecture microservices.
-  Long startup time
-  Large compute consumption
-  Poor performance

Traditional platform BPMN

-  A large monolith and Stateful
-  Not Optimized for cloud and architecture microservices.
-  Long startup time
-  Large compute consumption
-  Poor performance
-  Affect developers' productivity

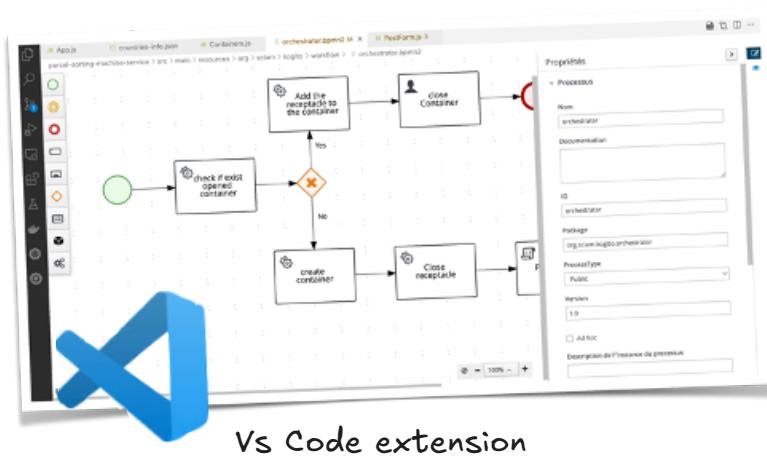
WHY KOGITO

- Cloud first priority
- Optimized for cloud architecture
- Multi runtime mode
- Technology Enabler
- Developer-centric Experience

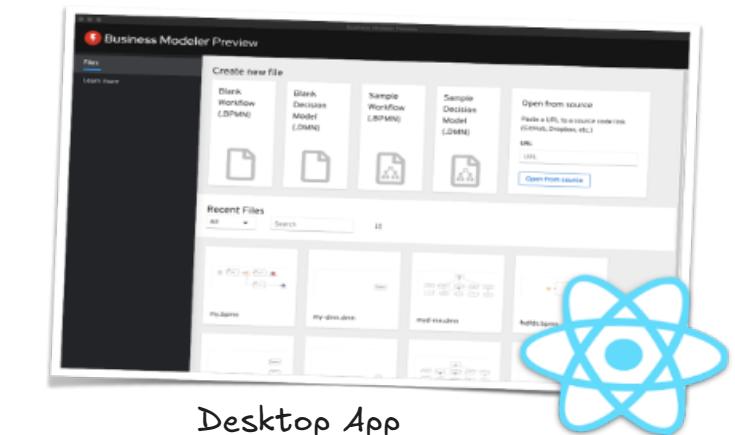
Kogito Capabilities

Domain-specific Service	Reactive Messaging	Data Index	Job Service	Management UI
 Decisions (DMN) Processes (BPMN) <ul style="list-style-type: none">◦ Service Tasks◦ Human Tasks◦ Gateways◦ Events◦ Timers/Jobs Test Scenarios Optaplannar Persistence Administration	 Apache kafka.	 Cloud Events Process + domain-specific events Data Grid Query <ul style="list-style-type: none">◦ GraphQL	 Jobs Timers Async	

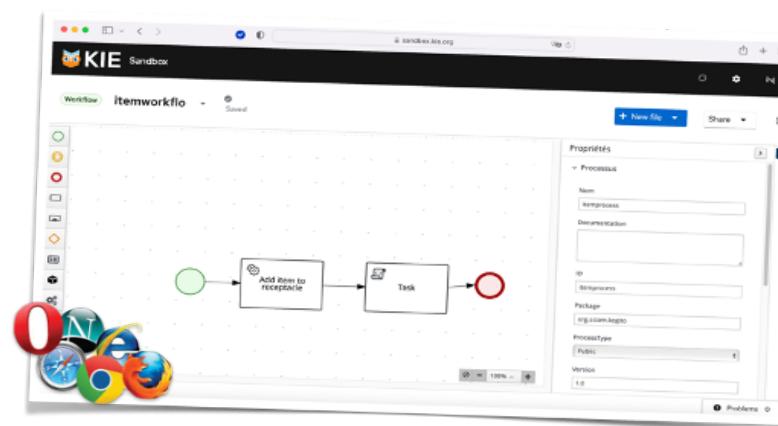
TOOLING



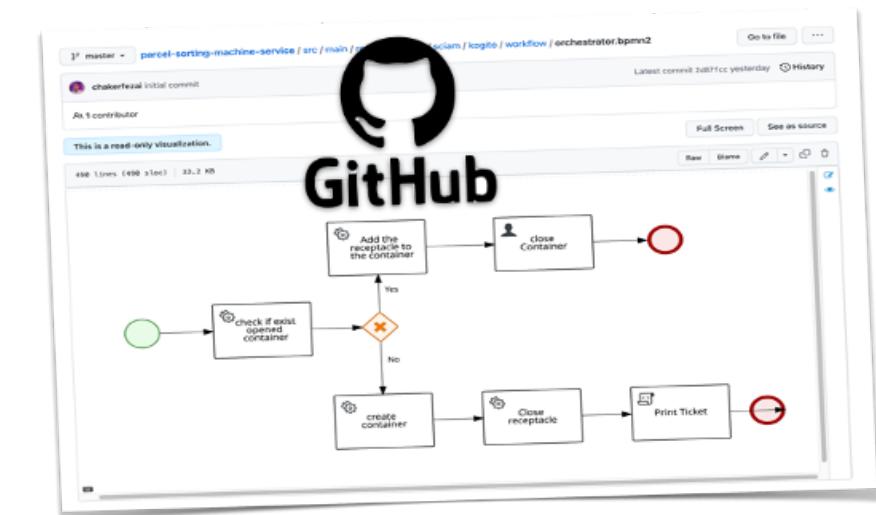
Vs Code extension



Desktop App



Online Editor

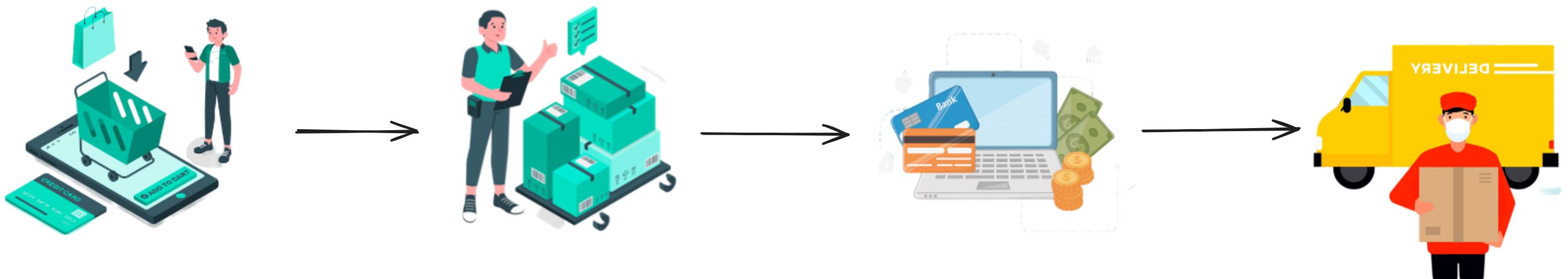


Github Chrome extension

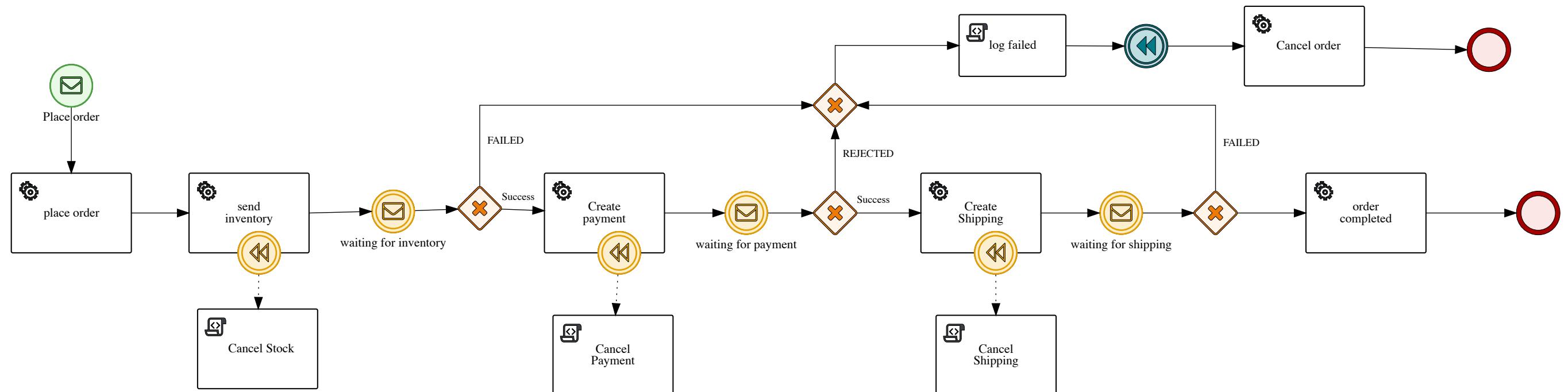
DEMO



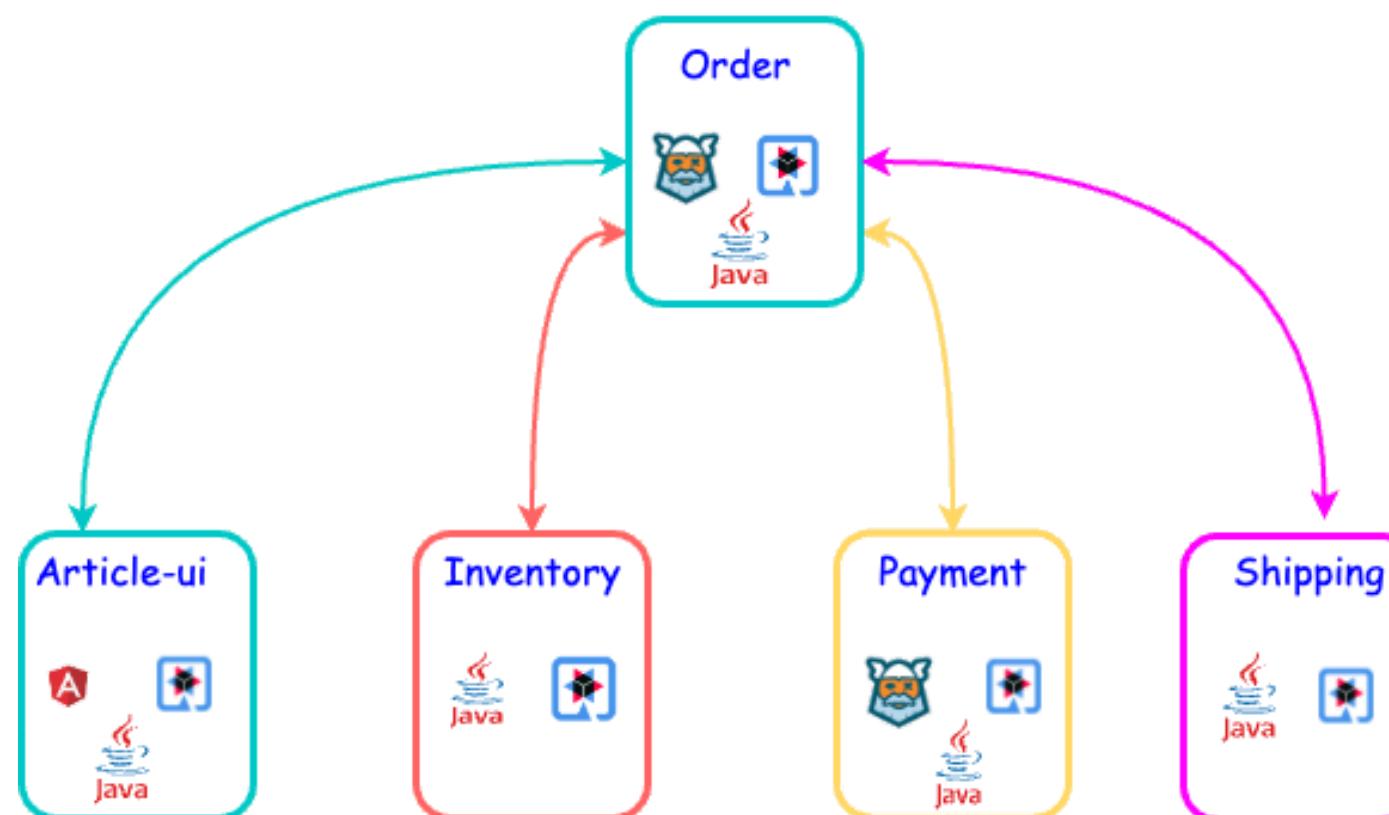
Order Shipment Workflow



Order Shipment Workflow BPMN 2.0



What you need for your first workflow



CODE GENERATION

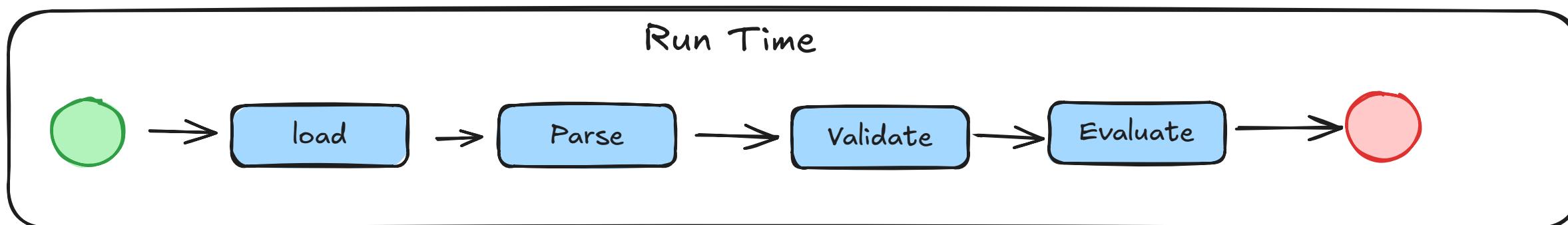
CODE GENERATION

Traditional Pipeline



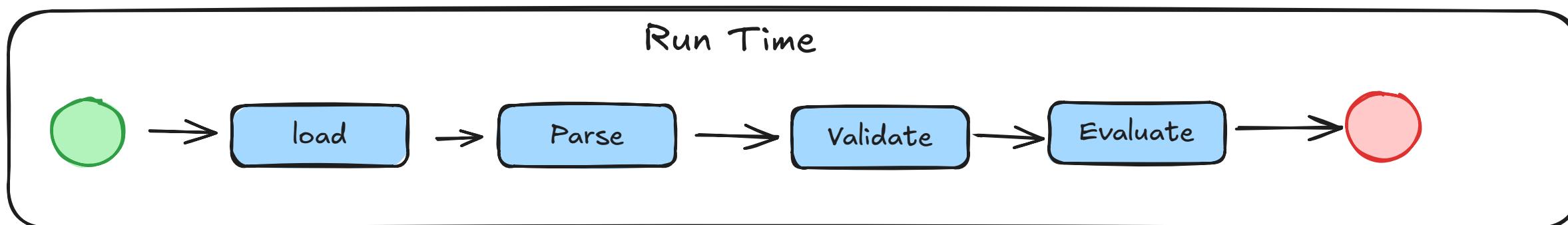
CODE GENERATION

Traditional Pipeline



CODE GENERATION

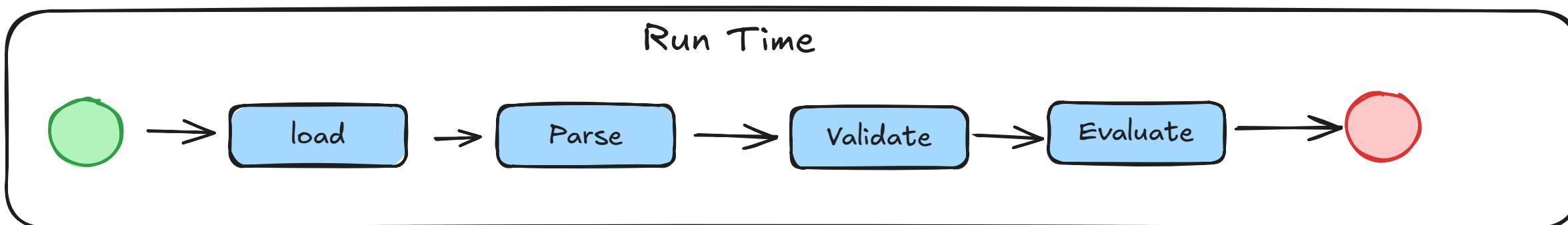
Traditional Pipeline



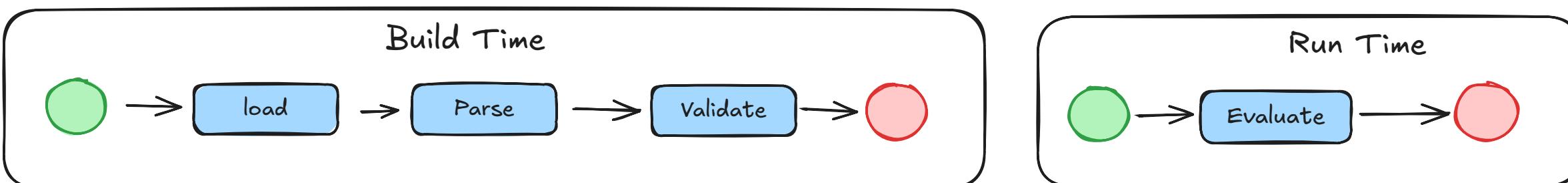
Kogito Pipeline

CODE GENERATION

Traditional Pipeline



Kogito Pipeline



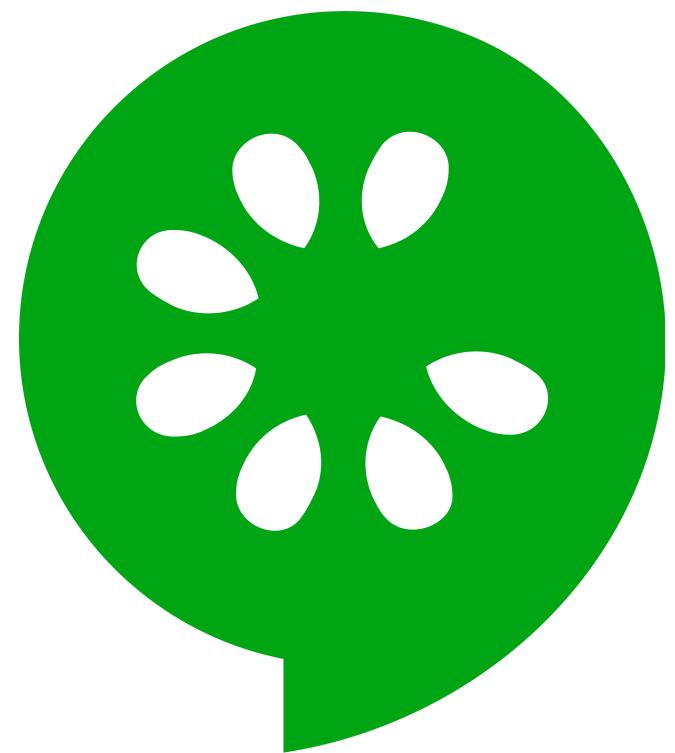
Challenges of Microservice Architecture

- Observability
- Scalability
- Easy to use / evolve
- High availability
- Schedules
- Human operation friendly
- Fault-Tolerance(Retry)
- Atomicity
- Data Consistency (Saga)
- Long Running Friendly



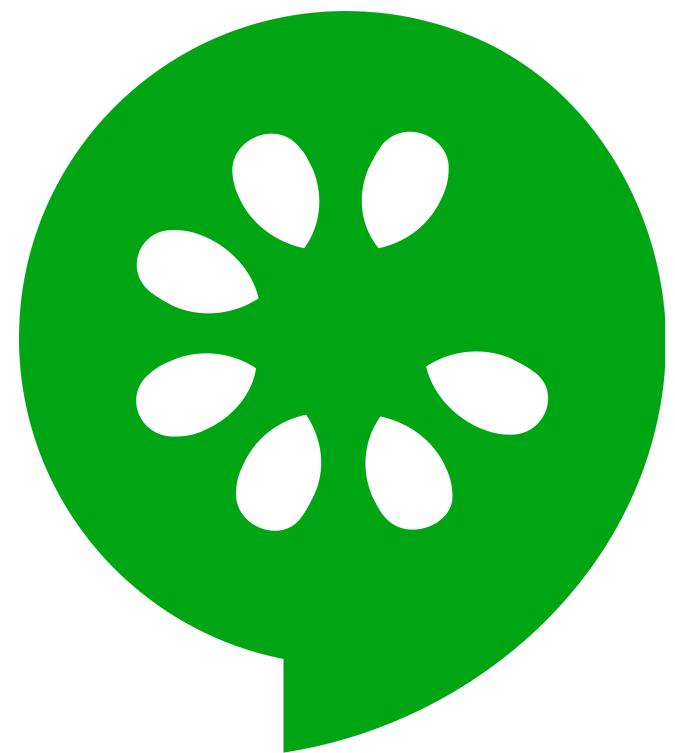
Other technologies used in the projet

Cucumber



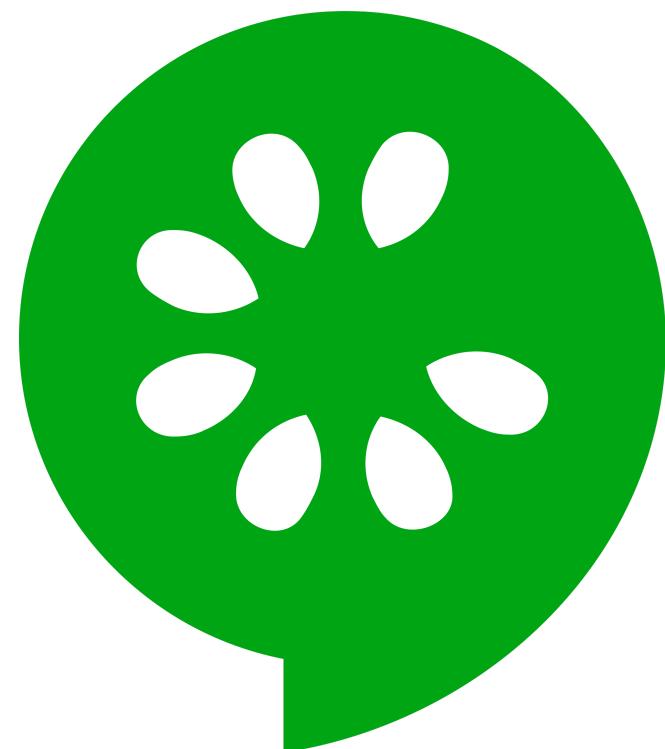
Cucumber

- Cucumber is a testing framework that supports Behavior Driven Development (BDD).



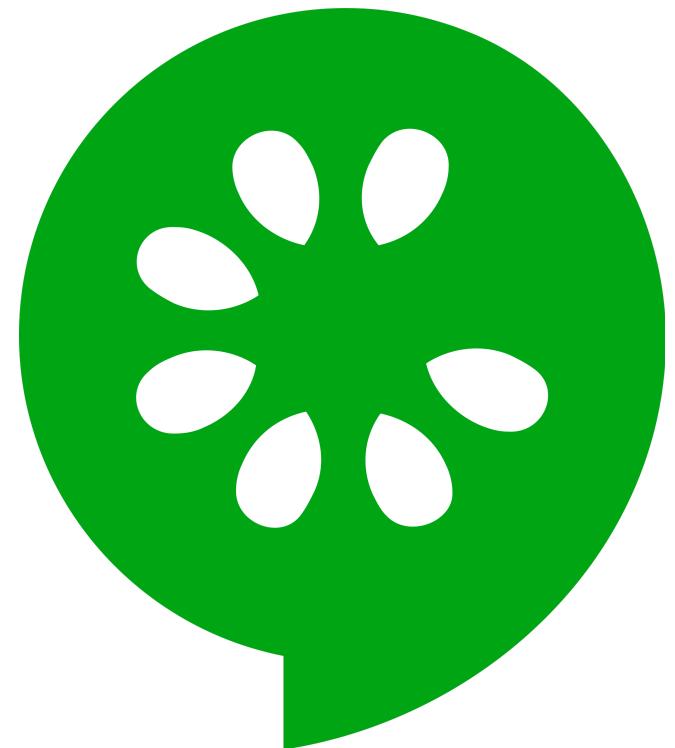
Cucumber

- Cucumber is a testing framework that supports Behavior Driven Development (BDD).
- As we have a lot of SQL queries to test, we needed a way to test them.



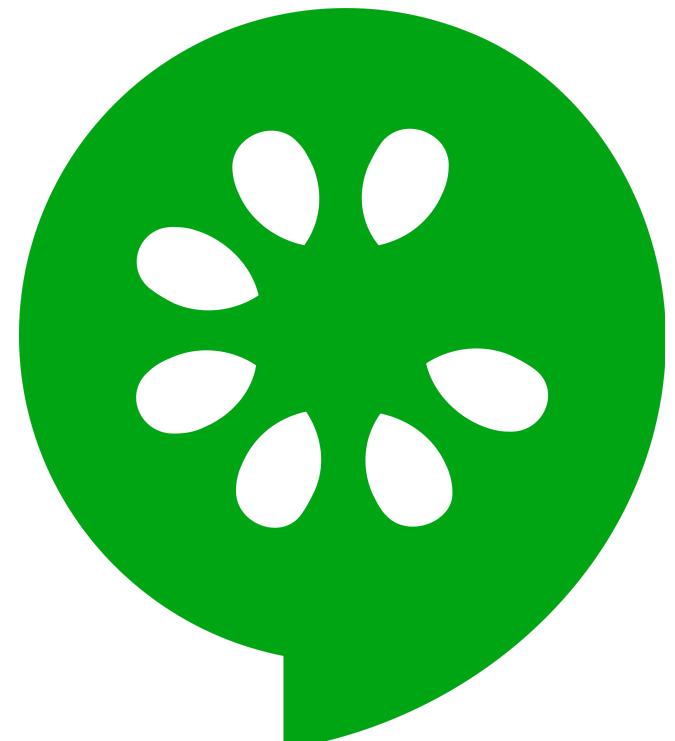
Cucumber

- Cucumber is a testing framework that supports Behavior Driven Development (BDD).
- As we have a lot of SQL queries to test, we needed a way to test them.
- It was a blessing for us to have a common language between devs and business.



Cucumber

- Cucumber is a testing framework that supports Behavior Driven Development (BDD).
- As we have a lot of SQL queries to test, we needed a way to test them.
- It was a blessing for us to have a common language between devs and business.
- In the end, BA were writing the tests and devs only creating new vocabulary.



Liquibase



Liquibase

- Liquibase is an open-source database schema change management tool.



Liquibase

- Liquibase is an open-source database schema change management tool.
- We have multiple schemas to manage on the projet



Liquibase

- Liquibase is an open-source database schema change management tool.
- We have multiple schemas to manage on the projet
- It was a bit hard to have our DBA accept this tool



Liquibase

- Liquibase is an open-source database schema change management tool.
- We have multiple schemas to manage on the projet
- It was a bit hard to have our DBA accept this tool
- Flyway was also considered but not chosen due to cost



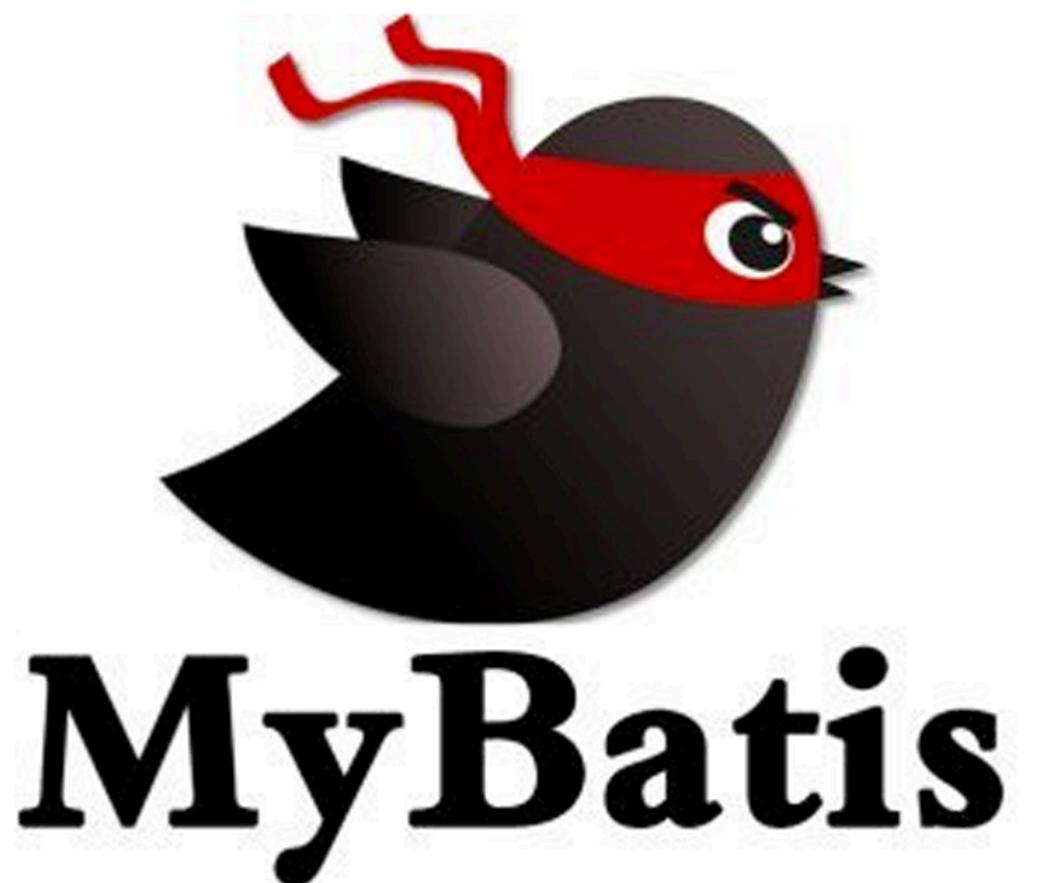
Mybatis



MyBatis

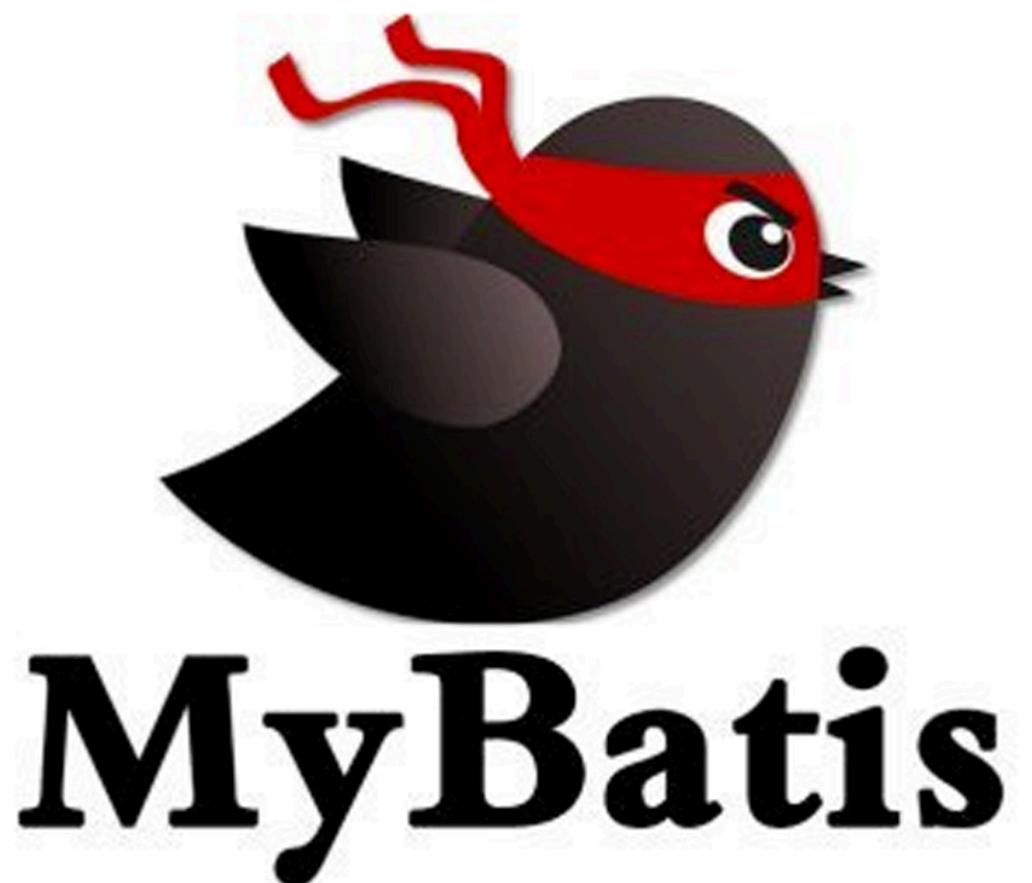
Mybatis

- Mybatis is a 20-year-old ORM Java framework.



Mybatis

- Mybatis is a 20-year-old ORM Java framework.
- It allows working very close to SQL.



Mybatis

- Mybatis is a 20-year-old ORM Java framework.
- It allows working very close to SQL.
- You can put dynamic part in your SQL queries.



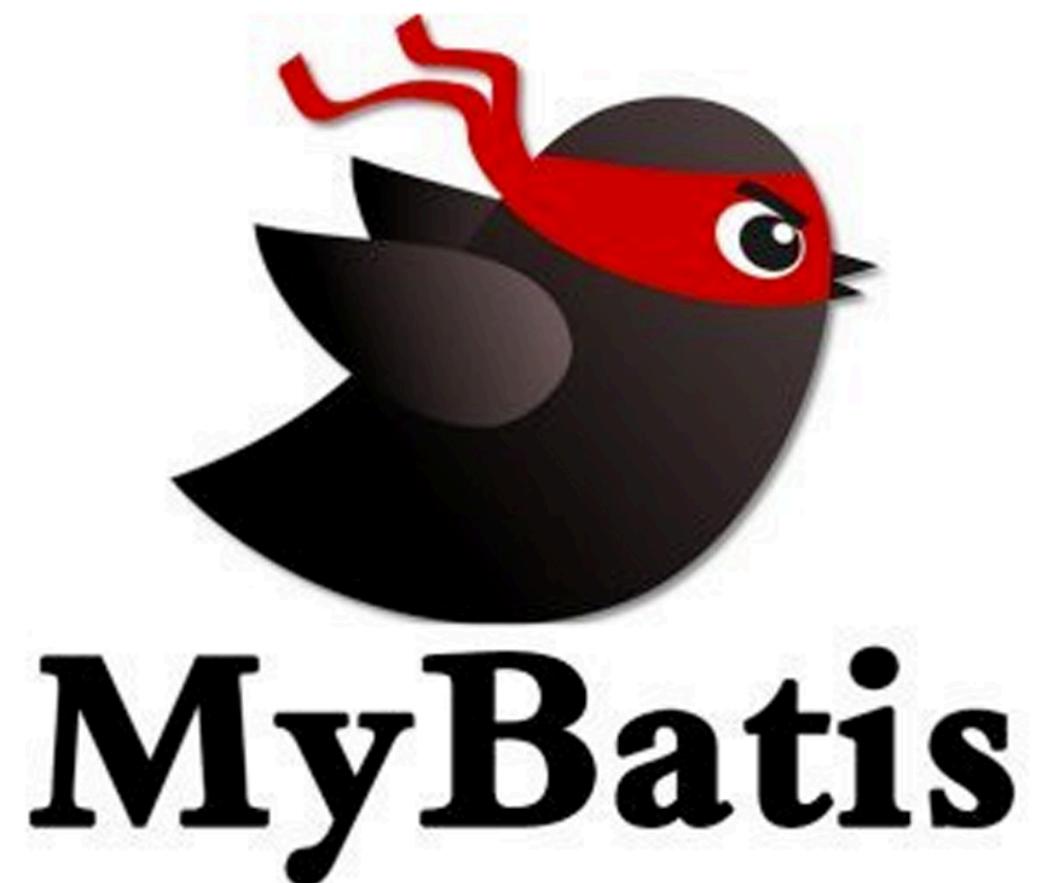
Mybatis

- Mybatis is a 20-year-old ORM Java framework.
- It allows working very close to SQL.
- You can put dynamic part in your SQL queries.
- Object Mapping is very flexible



Mybatis

- Mybatis is a 20-year-old ORM Java framework.
- It allows working very close to SQL.
- You can put dynamic part in your SQL queries.
- Object Mapping is very flexible
- Works with XML or Annotations



Monitoring and Observation UI



Global Dashboard View



Supervise the Global
State of the System

Global Dashboard View

- ⌚ Real-time global system overview



Supervise the Global
State of the System

Global Dashboard View

- ⌚ Real-time global system overview
- 📊 Statistics on incoming flows: total, in-progress, completed, failed



Supervise the Global
State of the System

Global Dashboard View

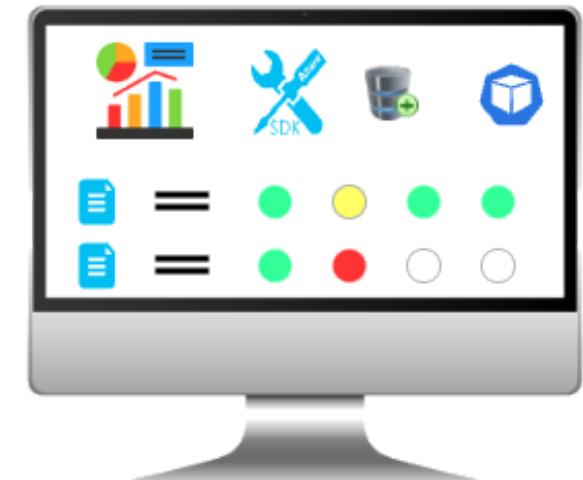
- ⌚ Real-time global system overview
- 📊 Statistics on incoming flows: total, in-progress, completed, failed
- 👁️ Supervise microservices and database health



Supervise the Global
State of the System

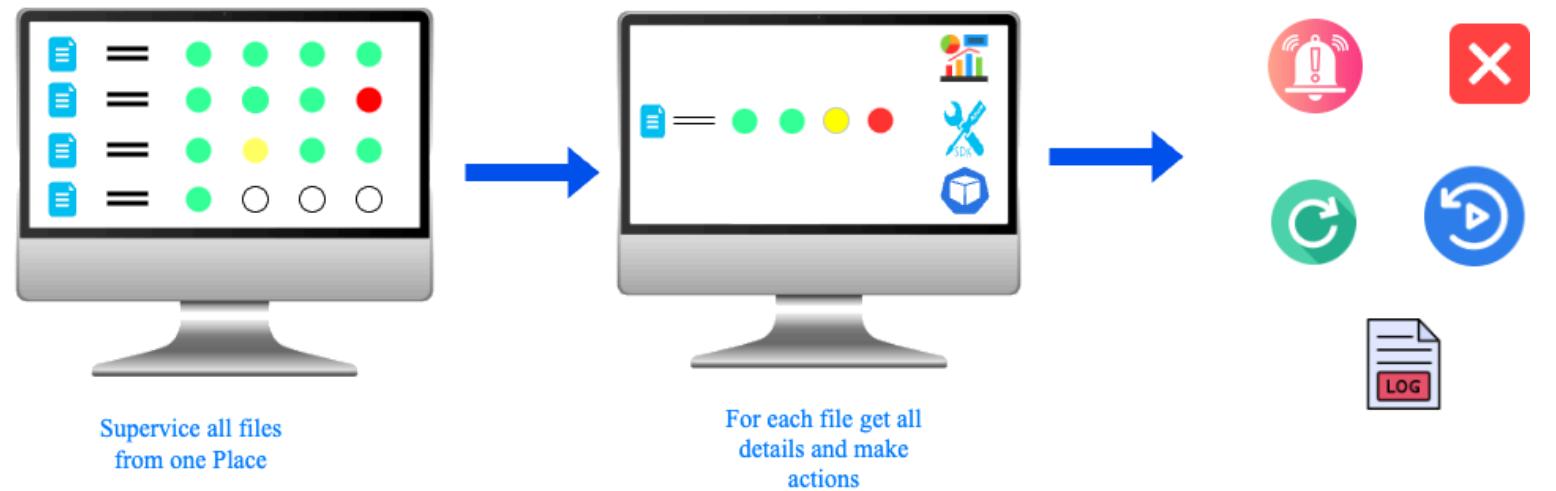
Global Dashboard View

- ⌚ Real-time global system overview
- 📊 Statistics on incoming flows: total, in-progress, completed, failed
- 👁️ Supervise microservices and database health
- 👤 Trigger corrective actions directly



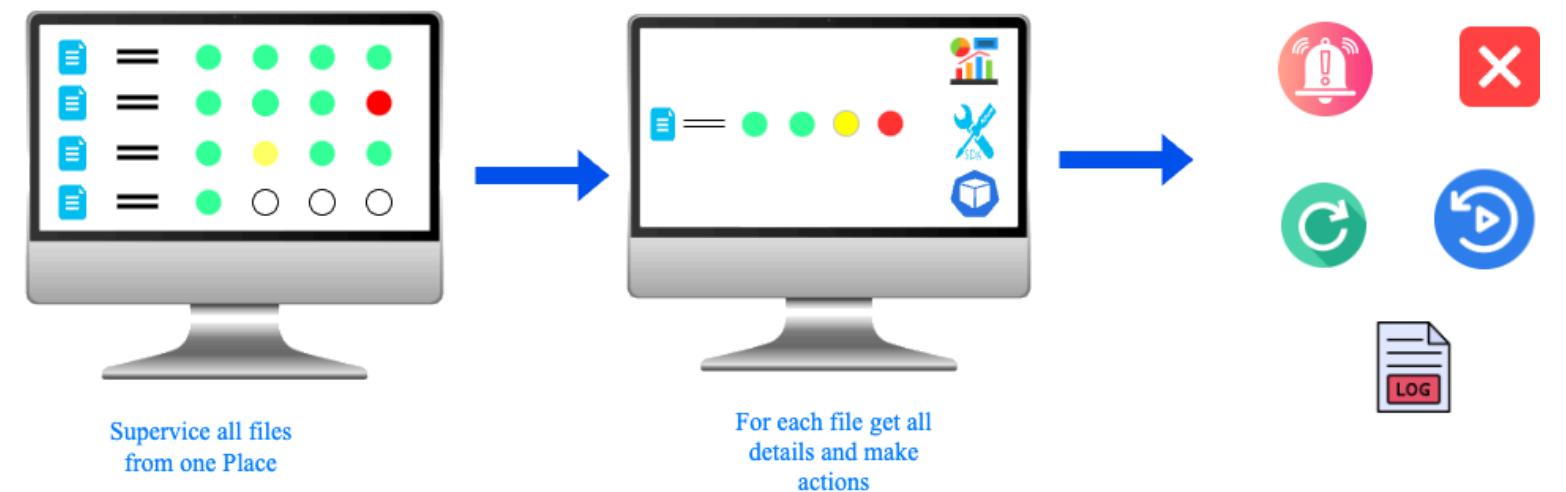
Supervise the Global
State of the System

Flow Details



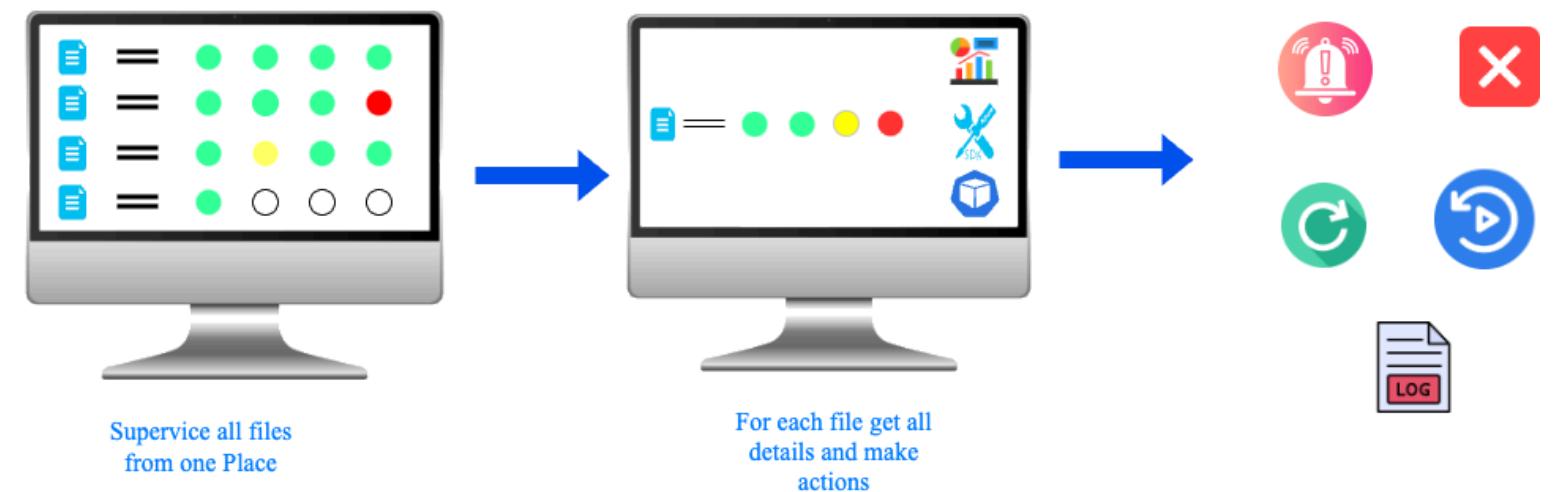
Flow Details

- ⌚ Filter flows by topology



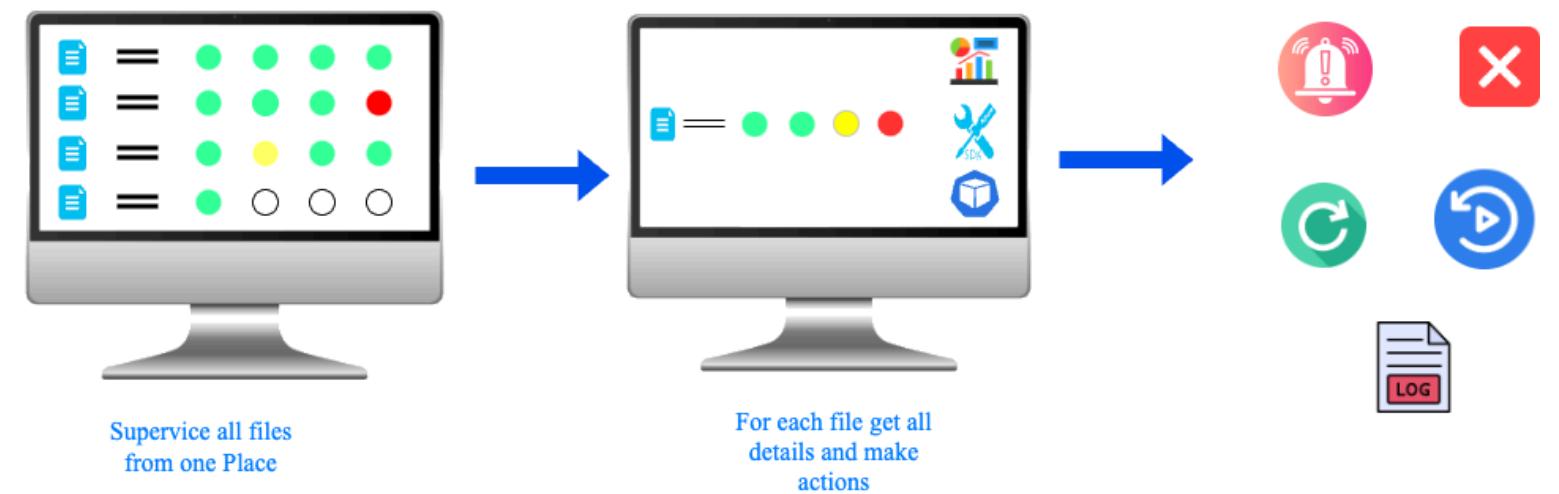
Flow Details

- ⌚ Filter flows by topology
- ⌚ Track step-by-step flow execution



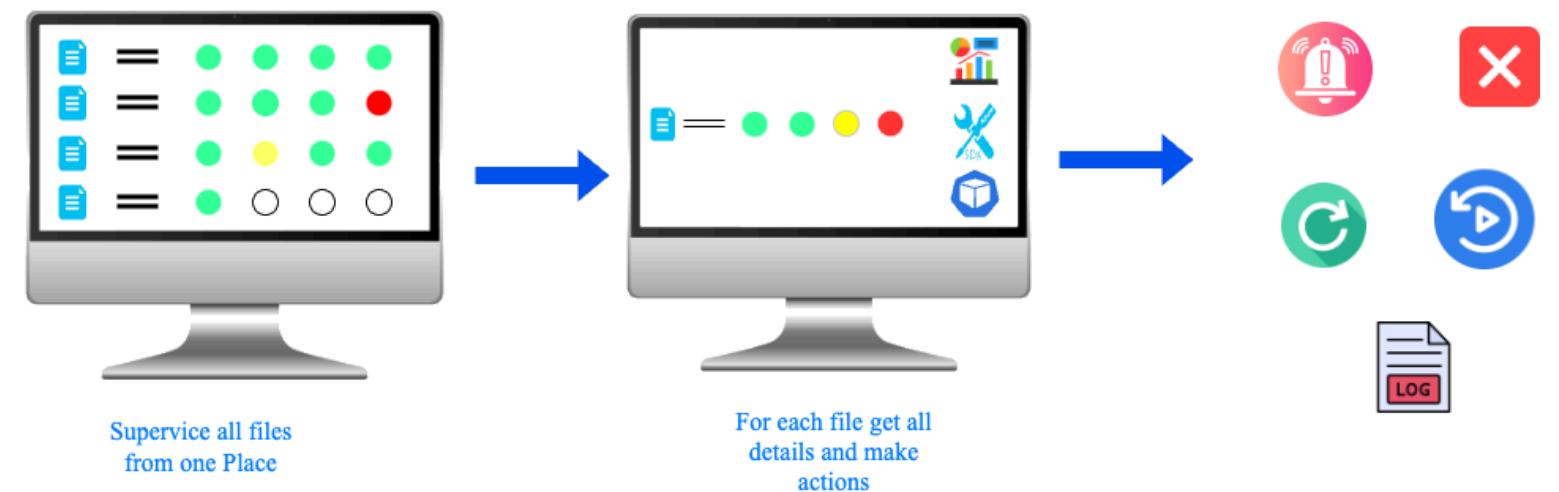
Flow Details

- ⌚ Filter flows by topology
- ⌚ Track step-by-step flow execution
- ⌚ Access step-level logs

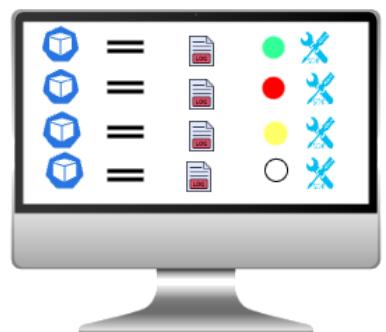


Flow Details

- ⌚ Filter flows by topology
- ⌚ Track step-by-step flow execution
- ⌚ Access step-level logs
- ⌚ Available actions:
 - Restart (from beginning or specific step)
 - Cancel
 - Temporarily disable a flow type



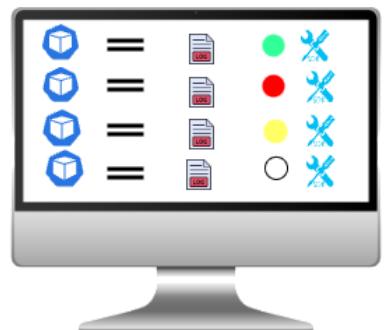
Microservices Management



microservices management

Microservices Management

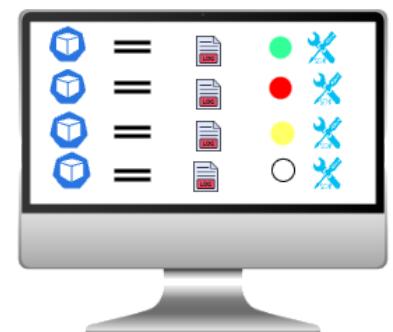
- List all pods per microservice



microservices management

Microservices Management

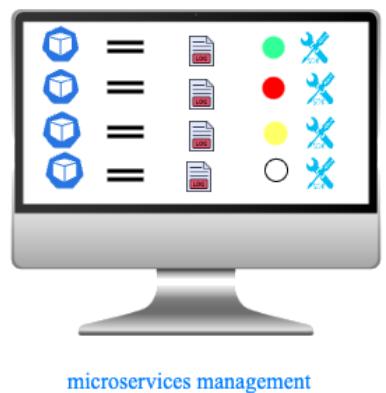
- List all pods per microservice
- Real-time status visualization



microservices management

Microservices Management

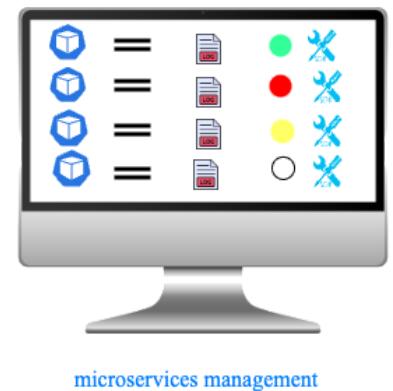
- List all pods per microservice
- Real-time status visualization
- Available actions:



microservices management

Microservices Management

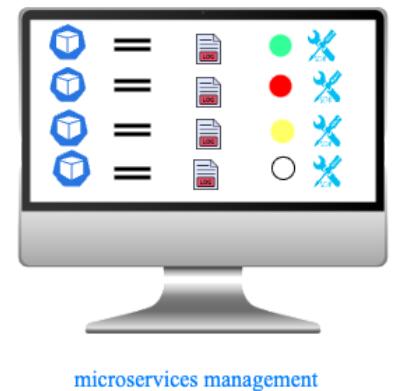
- List all pods per microservice
- Real-time status visualization
- Available actions:
- Restart pod



microservices management

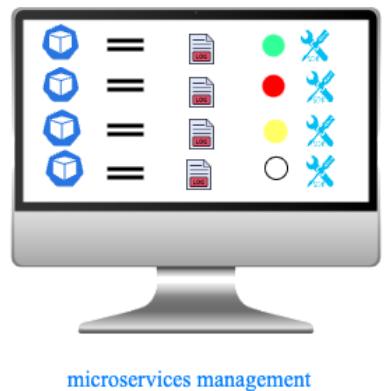
Microservices Management

- List all pods per microservice
- Real-time status visualization
- Available actions:
 - Restart pod
 - Manual stop



Microservices Management

- List all pods per microservice
- Real-time status visualization
- Available actions:
 - Restart pod
 - Manual stop



microservices management

```
public List<PodInfo> getPodsInfo() {  
  
    PodList kogitoPods = client.pods().inNamespace("kogito").list();  
    return kogitoPods.getItems().stream()  
        .map(pod -> {  
            String name = pod.getMetadata().getName();  
            String status = pod.getStatus() != null ? pod.getStatus().getPhase() : "Unknown"  
            return new PodInfo(name, status);  
        }).collect(Collectors.toList());  
}
```

Key Benefits

- Centralized system supervision
- Faster incident response
- Fine-grained system observability
- Reduced reliance on external tools and CLI



Questions



References

- *Slides are accessible here //TODO define slides url*
- *Slides source github.com/SCIAM-FR/mainframe-to-quarkus*
- *Slides generated with Asciidoctor and Reveal.js*

