D7.1. TECHNICAL SPECIFICATIONS AND DETAILED ARCHITECTURE REPORT

Grant Agreement:	833635
Project Acronym:	ROXANNE
Project Title:	Real time network, text, and speaker analytics for combating organised crime
Call ID: **Call name:**	H2020-SU-SEC-2018-2019-2020, Technologies to enhance the fight against crime and terrorism
Revision:	V1.0
Date:	18 December 2019
Due date:	28 February 2020
Deliverable lead:	AIRBUS
Work package:	WP7
Type of action:	RIA

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Disclaimer

The information, documentation and figures available in this deliverable are written by the “ROXANNE - ” Real time network, text, and speaker analytics for combating organised crime” project’s consortium under EC grant agreement 833635 and do not necessarily reflect the views of the European Commission.

The European Commission is not liable for any use that may be made of the information contained herein.

Copyright notice

Project co-funded by the European Commission within the H2020 Programme (2014-2020)		
Nature of deliverable:	R	
Dissemination Level		
PU	Public	☐
CO	Confidential, only for members of the consortium (including the Commission Services)	☒
EU-RES	Classified Information: RESTREINT UE (Commission Decision 2015/444/EC)	☐
* R: Document, report (excluding the periodic and final reports) DEM: Demonstrator, pilot, prototype, plan designs DEC: Websites, patents filing, press & media actions, videos, etc. OTHER: Software, technical diagram, etc.		

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

Revision history

Revision	Edition date	Author	Modified Sections / Pages	Comments
V0.1	10 December 2019	AIRBUS	All	Editing all the sections
V0.2	22 January 2020	AIRBUS	All	Editing all the sections
V0.3	29 January 2020	AEGIS	All	Add section 6 - Data visualisation framework description and comments throughout the document
V0.4	05 February 2020	AIRBUS	All	Editing all the sections
V0.4	19 February 2020	Phonexia	-	comments added
V0.4	20 February 2020	USAAR	Section 5	comments, questions and minor modifications
V0.4	24 February 2020	ITML	Section 4	Add an extra component (DFB) in the list which supports EDA/MSA architecture in ROXANNE; Add a detailed description of the component as a separate subsection (4.4).
V0.5	26 February 2020	AIRBUS	Section 2 Section 5 Section 8 Section 9	Update High level Architecture Schema Add places diarization service in videos Add a conclusion Add references
	27 February 2020	USAAR	Section 5	Addressed comments
	27 February 2020	IDIAP	All	Editing
	27 February 2020	SAIL	Section 5	Added comments
	28 February 2020	SAIL	All	
	28 February 2020	BUT	Section 5.2	Added this section.
	29 February 2020	LUH	Sections 5.7 - 5.10	Added sections for social network analysis services

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

	1 March 2020	UCSC	General check and sections 5.7 - 5.10	Checked the document for general consistency and clarity. Made some replacements (e.g. relationship —> relation) and corrected typos (“exemple”), edited sections 5.7 to 5.10
	02 March 2020	USAAR	Section 5, added additional NLP functionality	
	02 March 2020	SAIL	Section 5	Edited the section in general, inserted comments as footnotes, defined input/output structures for SD, ASR, LID, TD, SA services
1.0	05 March 2020	AIRBUS	All	First version release

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

Executive summary

ROXANNE, a novel platform combining advances of speech, language and video technologies and criminal network analysis for supporting investigators in their daily work especially on large criminal cases, has as its main goal to speed up the investigative processes, as well as to reduce the cost and burden to the society caused by organized crime activities. ROXANNE focuses on typical investigation processes where a significant amount of information is collected from telecommunication sources (e.g. wiretaps, interview recordings or audio provided by social media, complemented by video and geographical information). Usually two particular but separated approaches are employed in practice by LEAs: (i) identification of individuals from media (audio, video) by means of speaker identification, and (ii) analysis of relations among individuals and the whole structure of criminal network derived from various kinds of police data (i.e. criminal network analysis). The main objective of ROXANNE is to develop novel statistical methods and a platform to interface these two approaches (i.e. person identification and criminal network analysis technologies) to substantially increase their performance which will naturally lead to better investigation and identification capabilities of LEAs. ROXANNE also includes multilingual automatic speech recognition, natural language processing, video analysis and relation analysis in extraction of information from the available data.

To achieve these goals, ingest and process huge volumes of heterogeneous data, and get fine and useful information a highly available and resilient platform is needed. This document is describing the ROXANNE Platform Architecture.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

1. Introduction

1.1. Purpose of the document

This document is the first deliverable of WP7 : D7.1 Technical specifications and detailed architecture report (M6, leader: AIRBUS); it describes the technical specifications and the design architecture of the platform and integration framework given the requirements.

1.2. Document structure

Section 2 presents the high-level architecture used in ROXANNE project to ingest data, orchestrate its processing and then visualise results efficiently.

Section 3 introduces main principles of Event Driven Microservices Architecture chosen to answer to ROXANNE's challenges .

Section 4 describes software components and frameworks that are used to implement this architecture.

Section 5 defined interfaces, inputs and outputs of all ROXANNE processing components.

Section 6 finally describes the data visualisation framework used in ROXANNE to present data and processing results an efficiently to end users.

1.3. Technical definitions

This paragraph contains some technical definitions that will be used in this deliverable and all along the project.

API – a set of protocols, routines and tools for building software and applications. The purpose of an API is to express the functionalities of a software in terms of the interface (e.g., operations, inputs, outputs, underlying types, etc.) allowing for changes in implementation without compromising the interface itself. API can be a simple specification of remote calls exposed to consumers (e.g., SOAP and REST services), a library (e.g., specifications for routines, classes, variables, etc.), a set of classes with associated list of class methods (e.g., documentation of all the kinds of objects one can derive from the class definitions, and their associated possible behaviours).

Note: API is associated with both frameworks and libraries. Certain behaviour can be implemented by a library or built into a framework, in either case, described by an API.

Framework – a universal and reusable software environment that provides a specific functionality supposed to be part of a larger software application. The purpose of a framework is to ease the development of software applications, products and solutions. Frameworks provide a general solution that can be made more specific by the end-user by adding some user-written code.

Platform (specifically, the ROXANNE Platform) – The software which will be developed as an outcome of the ROXANNE project, which is an extensible analytics platform to support LEAs with advanced investigation capabilities, combining speech, language and video technologies with criminal network analysis.

Component – Each one of the interacting parts of the ROXANNE platform which is responsible for the processing of different data types, e.g., Speech and speaker analytics component, as well as visualising outputs in a user-friendly way.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Microservice – Way to develop each ROXANNE component in an independent and loosely coupled process

Technology/ies – The element of a component which is responsible for carrying out a specific function, e.g., Speaker identification.

Containers – A standard unit of software that packages up code and all its dependencies and libraries so the application runs quickly and securely by taking advantage of Operation-system-level virtualization. All ROXANNE components are packaged in containers.

Orchestrator – Tool used to orchestrate containers: deploy, start, stop, scale containers inside the ROXANNE platform.

2. ROXANNE High Level Architecture

ROXANNE high-level architecture is the following:

- During data ingestion, 3 processes are running at the same time
  - Pushing raw data in Ceph object storage
  - Pushing corresponding metadata in a Kafka topic: filename, URI to raw data on object storage, date…
  - Consuming Kafka's queue (topic) for storing metadata in the Case Management System
- Nifi is used for data processing orchestration:
  - By listening to Kafka input topics
  - By calling REST microservices to process / enrich data
  - By finally pushing processed data in Elasticsearch indices, which is periodically retrieved for visualisation purposes
- All components are running as containers in a Kubernetes cluster
- Stateful components are running as statefulset kubernetes components (always up, resilient data, stable network identities):
  - Apache Nifi
  - Apache Kafka
  - Elasticsearch master / data nodes
- Processing Lightweight Microservices are running as Kubernetes workload
  - Auto scaling
  - Ephemeral
  - Resources sharing (CPU, RAM, GPU, Data volumes, Object storage)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

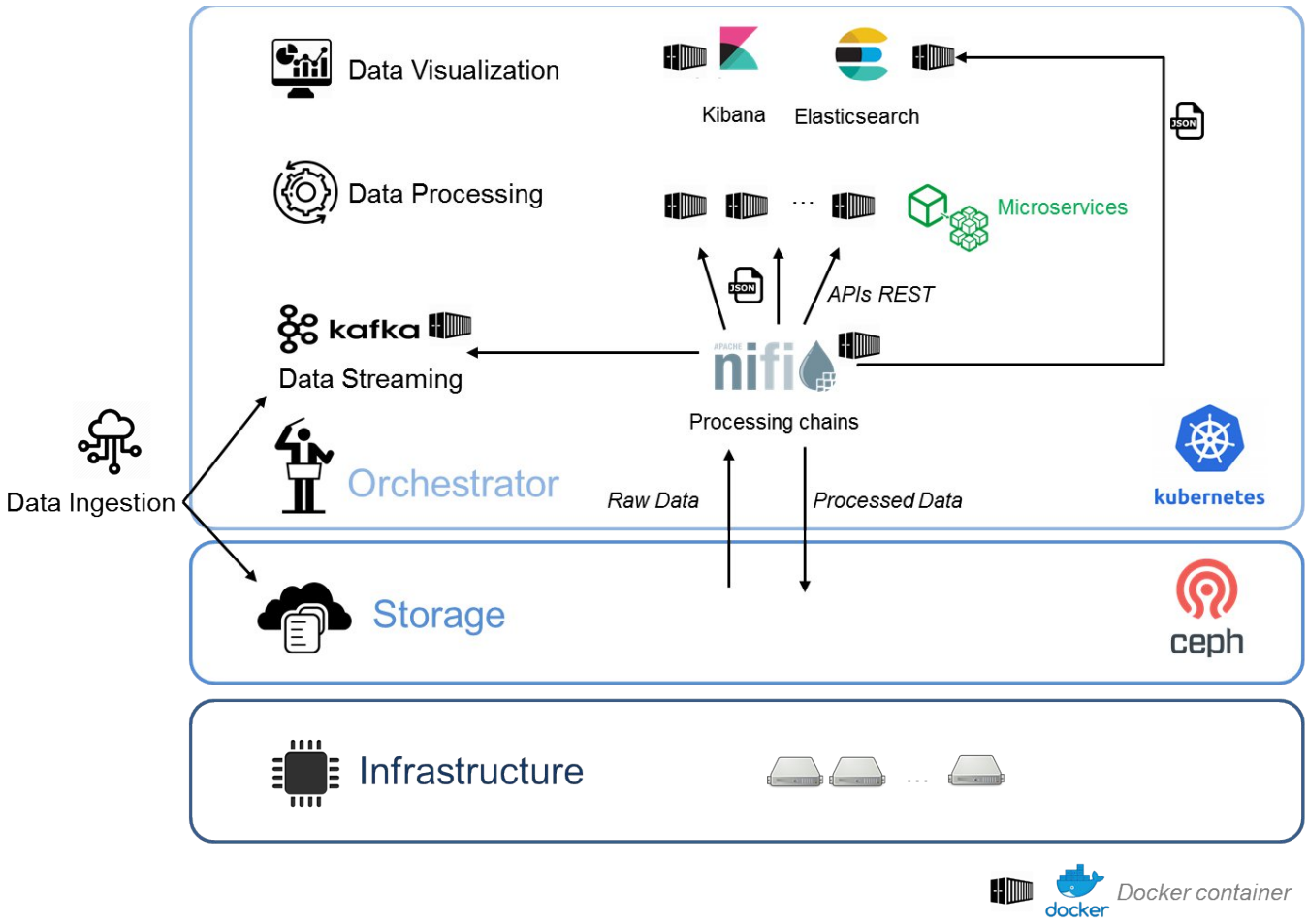


Figure 1 - ROXANNE High Level Architecture

3. Event driven Microservices Architecture

In traditional Service Oriented Architecture (SOA), elements could be developed relatively autonomously but must be coordinated with others to fit into the overall design.

During the last years new approaches have appeared in terms of applications developments:

- Microservices
- Event-driven architecture

ROXANNE European Project analytics is using such architecture.

3.1 Microservices

Microservices is a software architecture paradigm from which a complex set of applications is broken into several independent and loosely coupled processes, often specialized in a single task. Independent processes communicate with each other using language-agnostic APIs.

Advantages:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

In Micro Service Architecture (MSA), developers can create and activate new Microservices without prior coordination with others. Their adherence to MSA principles makes continuous delivery of new, or modified, services possible.

Microservices offers the following:

- Agility: separated teams could develop separate services
- Scalability: processing tasks are automatically load-balanced between several services instances
- Easiest deployment: Each service is self-sufficient in its own container and will not affect other services.

3.2 Event Driven Architecture

Event-Driven Architecture (EDA), is a software architecture pattern promoting the production, detection, consumption of, and reaction to “events”.

An event can be defined as "a significant change in state".

Building systems around an Event-Driven Architecture simplifies horizontal scalability in distributed computing models and makes them more resilient to failure.

In EDA the coupling between services is loose coupling and the communications are all asynchronous.

Main characteristics of EDA:

- Strong internal consistency (using a pivot exchange format, JSON, XML...)
- Low external linkages (through the use of events)
- Opposite to SOA where a "supplier" makes a service answering a consumer’s request...
- ...a "service" warns by sending an event that it has performed a given operation.
- It is up to potential customers to handle this event.

Advantages:

- Simplifies horizontal scalability
- More resilient to failure
- Adding extra nodes becomes trivial as well

Disadvantages:

- Could increase complexity
- We don’t always know exactly what components are part of the system and the dependencies between them

A service can:

- Be coded in any language,
- Run on any platform (hardware and software).

A service must:

- Subscribe to the events it wants to treat
- Process the events to which it subscribes without prejudging any order and emit an event report of the action it has just completed
- Provide the events it is likely to emit whose structures are published,
- Be autonomous (to have all the information necessary for its execution: no notion of state)
- Comply with a set of contracts (operating rules).

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

4. Technical Frameworks

To implement such EDA / MSA architecture in ROXANNE we chose to use following frameworks / components:

- Docker
- Apache Kafka
- Apache Nifi
- Kubernetes
- Data Fusion Bus (DFB)

4.1 Docker

Docker provides a way to run applications securely isolated in a container, packaged with all its dependencies and libraries. It performs operation-system-level virtualization, also known as "containerization".

Containers are:

- Flexible: Even the most complex applications can be containerized.
- Lightweight: Containers leverage and share the host kernel.
- Interchangeable: You can deploy updates and upgrades on-the-fly.
- Portable: You can build locally, deploy to the cloud, and run anywhere.
- Scalable: You can increase and automatically distribute container replicas.
- Stackable: You can stack services vertically and on-the-fly.

A container runs natively on Linux and shares the kernel of the host machine with other containers.

It runs a discrete process, taking no more memory than any other executable, making it lightweight.

By contrast, a virtual machine (VM) runs a full-blown “guest” operating system with virtual access to host resources through a hypervisor. In general, VMs provide an environment with more resources than most applications need.

In ROXANNE we choose to put each Microservice in a container.

4.2 Apache Kafka

Apache Kafka is a distributed streaming platform. It allows writing scalable stream processing applications that react to events in real-time.

A distributed streaming platform:

- Publish and subscribe to streams of records, similar to a message queue or enterprise messaging system
- Store streams of records in a fault-tolerant durable way
- Process streams of records as they occur

Kafka is generally used for two broad classes of applications:

- Building real-time streaming data pipelines that reliably get data between systems or applications
- Building real-time streaming applications that transform or react to the streams of data

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

Kafka runs as a cluster on one or more servers that can span multiple data centers.The Kafka cluster stores streams of records in categories called topics.

Each record consists of a key, a value, and a timestamp.

Apache Kafka – 4 APIs:

- The Producer API allows an application to publish a stream of records to one or more Kafka topics.
- The Consumer API allows an application to subscribe to one or more topics and process the stream of records produced to them.
- The Streams API allows an application to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams.
- The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.

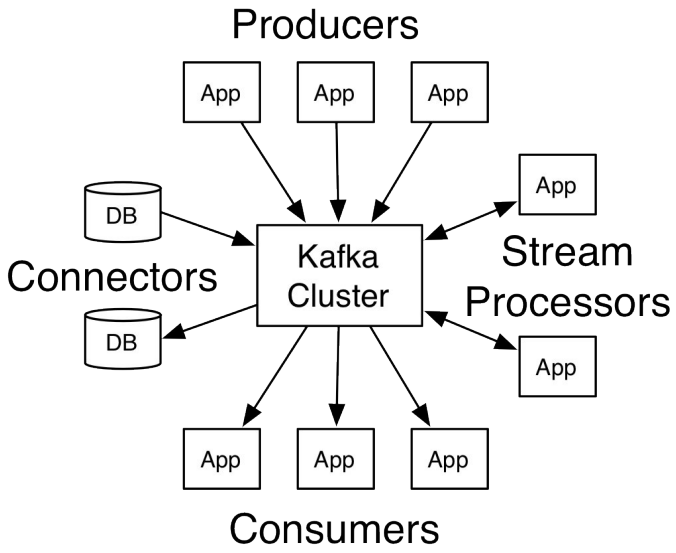


Figure 2 - Kafka Cluster

Topics:

Kafka topics are divided into a number of partitions. Each partition is an ordered, immutable sequence of records that is continually appended to a structured commit log.

Always multi-subscriber; that is, a topic can have zero, one, or many consumers that subscribe to the data written to it.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

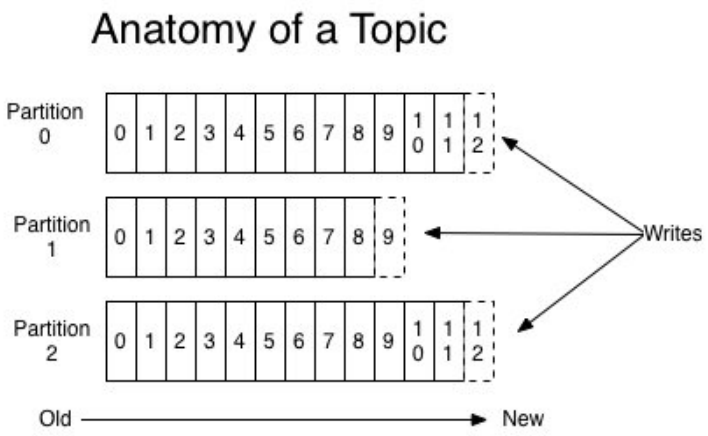


Figure 3 - Kafka Topic

Consumers:

- Only metadata retained on a per-consumer basis is the offset or position of that consumer in the log
- Offset is controlled by the consumer
- Normally a consumer will advance its offset linearly as it reads records
- A consumer can reset to an older offset to reprocess data from the past or skip ahead to the most recent record and start consuming from "now"

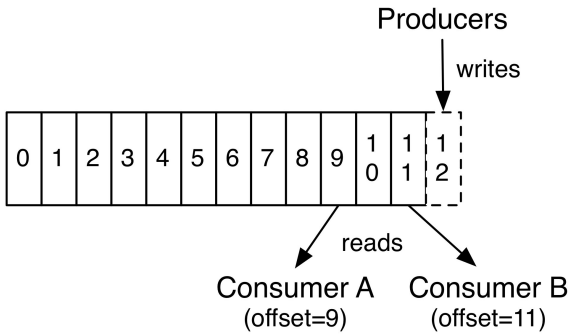


Figure 4 - Kafka Consumers

Groups:

- Consumers label themselves with a consumer group name
- Each record published to a topic is delivered to one consumer instance within each subscribing consumer group
- Consumer instances can be in separate processes or on separate machines

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

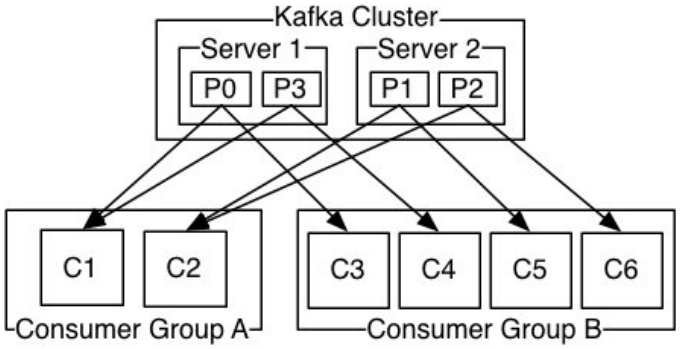


Figure 5 – Kafka Consumer groups

4.2.1 Kafka as a Messaging System

Messaging traditionally has two models: queuing and publish-subscribe.

- In a queue, a pool of consumers may read from a server and each record goes to one of them.
- In publish-subscribe, the record is broadcasted to all consumers.

Each of these two models has a strength and a weakness.

The strength of queuing is that it allows you to divide up the processing of data over multiple consumer instances, which lets you scale your processing.

Unfortunately, queues are not multi-subscriber—once one process reads the data it's gone.

Publish-subscribe allows you to broadcast data to multiple processes but has no way of scaling processing since every message goes to every subscriber.

The consumer group concept in Kafka generalizes these two concepts.

As with a queue the consumer group allows you to divide up processing over a collection of processes (the members of the consumer group).

As with publish-subscribe, Kafka allows to broadcast messages to multiple consumer groups.

The advantage of Kafka's model is that every topic has both these properties—it can scale processing and is also multi-subscriber—there is no need to choose one or the other.

4.3 Apache Nifi

Apache Nifi is a powerful and reliable system to process and distribute data. It is designed to automate the flow of data between software systems. It supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic.

It is based on the “NiagaraFiles” software previously developed by the NSA and was open-sourced in 2014.

Some of the high-level capabilities of Apache Nifi include:

- A web user interface
  - Seamless experience between design, control, feedback and monitoring
- Highly configurable
  - Loss tolerant vs guaranteed delivery

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

  - Low latency vs high throughput
  - Dynamic prioritization
  - Flow can be modified at runtime
  - Back pressure
- Data provenance
  - Track data-flow from beginning to end
- Designed for extension
- Secure
  - SSL, SSH, HTTPS, encrypted content…
  - Multi-tenant authorization and internal authorization/policy management

In Microservices architecture the data is the contract between the loosely coupled services. Nifi is a robust way to route data between those services.

4.3.1 Apache Nifi components

In Nifi, black boxes called processors exchange chunks of information named FlowFiles through queues that are named connections. Finally, the FlowFile Controller is responsible for managing the resources between those components.

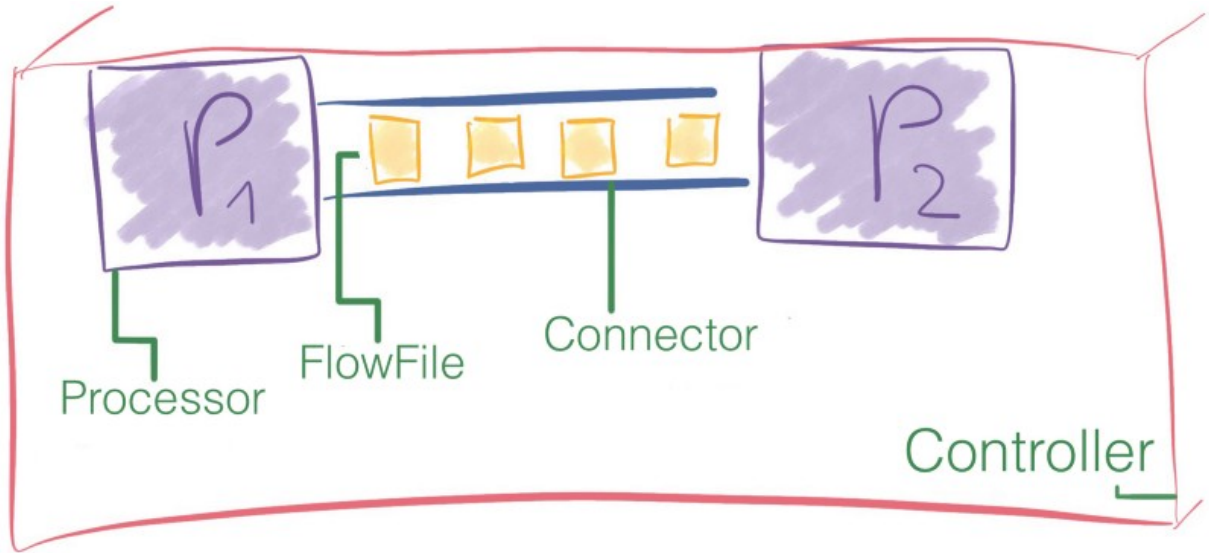


Figure 6 - Nifi main concepts

FlowFilehttps://www.docker.com/

In NiFi, the **FlowFile** is the information packet moving through the processors of the pipeline.

A **FlowFile** comes in two parts:

- Attributes, which are key/value pairs. For example, the file name, file path, and a unique identifier are standard attributes.
- Content, a reference to the stream of bytes compose the FlowFile content.

The **FlowFile** does not contain the data itself. That would severely limit the throughput of the pipeline.

Instead, a **FlowFile** holds a pointer that references data stored at someplace in the local storage. This place is called the **Content Repository**.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

FlowFile Processor

A **Processor** is a black box that performs an operation. **Processors** have access to the attributes and the content of the **FlowFile** to perform all kind of actions. They enable ROXANNE to perform many operations in data ingress, standard data transformation/validation tasks, and saving this data to various data sinks.

NiFi provides many **Processors** out of the box (293 in Nifi 1.9.2). Nifi also offers the possibility to write and use new custom **Processors**.

Process Group

A bunch of **processors** put together with their connections can form a process group. You add an input port and an output port so it can receive and send data.

Content Repository

To access the content, the **FlowFile** claims the resource from the **Content Repository**. The latter keeps tracks of the exact disk offset from where the content is and streams it back to the **FlowFile**.

Not all **processors** need to access the content of the **FlowFile** to perform their operations — for example, aggregating the content of two **FlowFiles** doesn't require to load their content in memory.

When a processor modifies the content of a **FlowFile**, the previous data is kept. NiFi copies-on-write, it modifies the content while copying it to a new location. The original information is left intact in the **Content Repository**.

FlowFile Repository

The attributes of all the **FlowFiles** currently in use, as well as the reference to their content, are stored in the **FlowFile Repository**.

At every step of the pipeline, a modification to a **FlowFile** is first recorded in the **FlowFile Repository**, in a write-ahead log, before it is performed.

For each **FlowFile** that currently exist in the system, the **FlowFile Repository** stores:

•The **FlowFile** attributes

•A pointer to the content of the **FlowFile** located in the **FlowFile Repository**

•The state of the **FlowFile**. For example: to which queue does the **Flowfile** belong at this instant.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

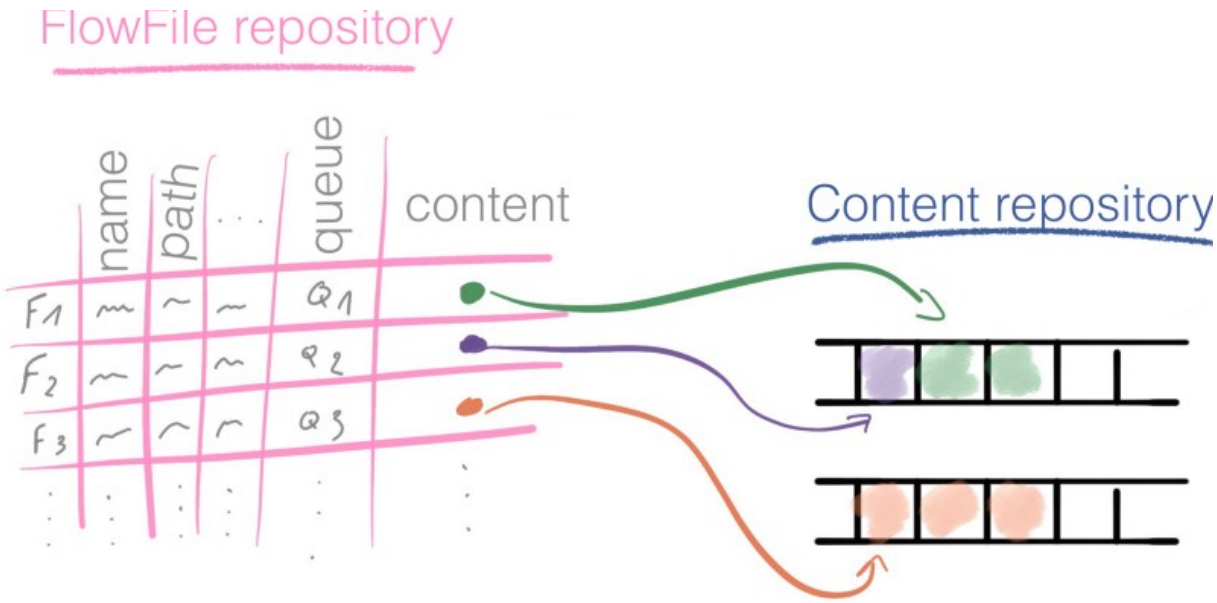


Figure 6 - Nifi FlowFile & Content Repositories

Provenance repository

Every time a **FlowFile** is modified, NiFi takes a snapshot of the **FlowFile** and its context at this point. The name for this snapshot in NiFi is a Provenance Event. The **Provenance Repository** records Provenance Events.

The idea behind the FlowFile Repository and the Provenance Repository is quite similar, but they don't address the same issue.

- The **FlowFile Repository** is a log that contains only the latest state of the in-use **FlowFiles** in the system. It is the most recent picture of the flow and makes it possible to recover from an outage quickly.
- The **Provenance Repository**, on the other hand, is more exhaustive since it tracks the complete life cycle of every **FlowFile** that has been in the flow.

On top of offering the complete lineage of the data, the **Provenance Repository** also offers to replay the data from any point in time.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

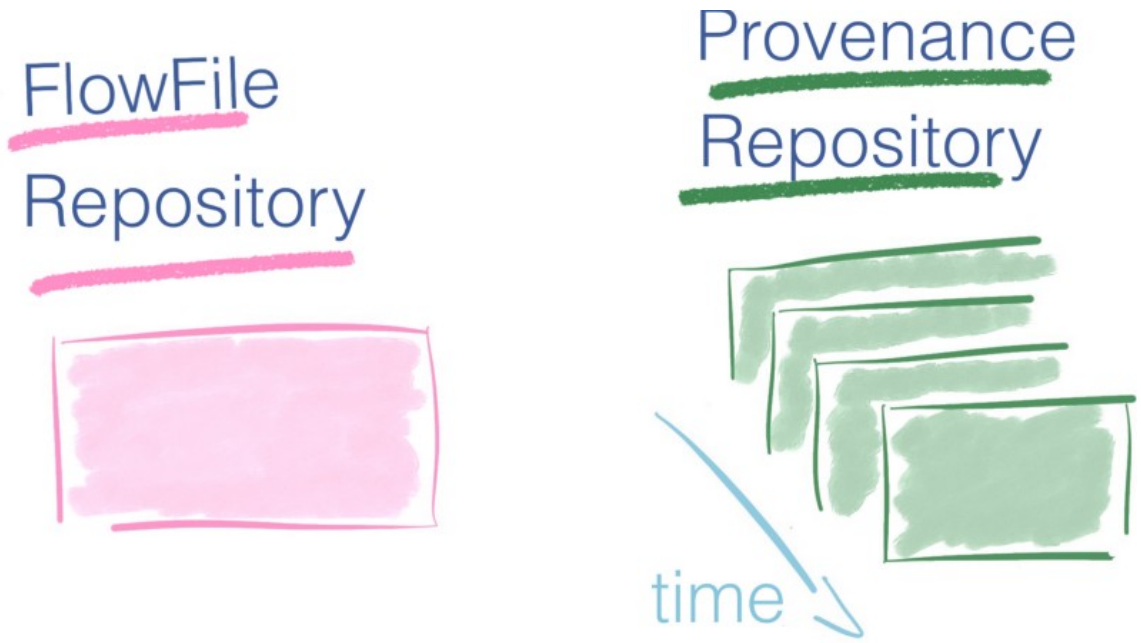


Figure 7 - Nifi FlowFile vs Provenance Repositories

Connections

Connections are the queues between processors. These queues allow **processors** to interact at differing rates. **Connections** can have different capacities like there exist different size of water pipes.

Because **processors** consume and produce data at different rates depending on the operations they perform, **connections** act as buffers of **FlowFiles**. There is a limit on how many data can be in the **connection**.

If the number of **FlowFiles** or the quantity of data goes above the defined threshold, **backpressure** is applied. The Flow Controller won't schedule the previous **processor** to run again until there is room in the queue.

When the number of **FlowFiles** or the associated data go beyond the threshold, a swap mechanism is triggered.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

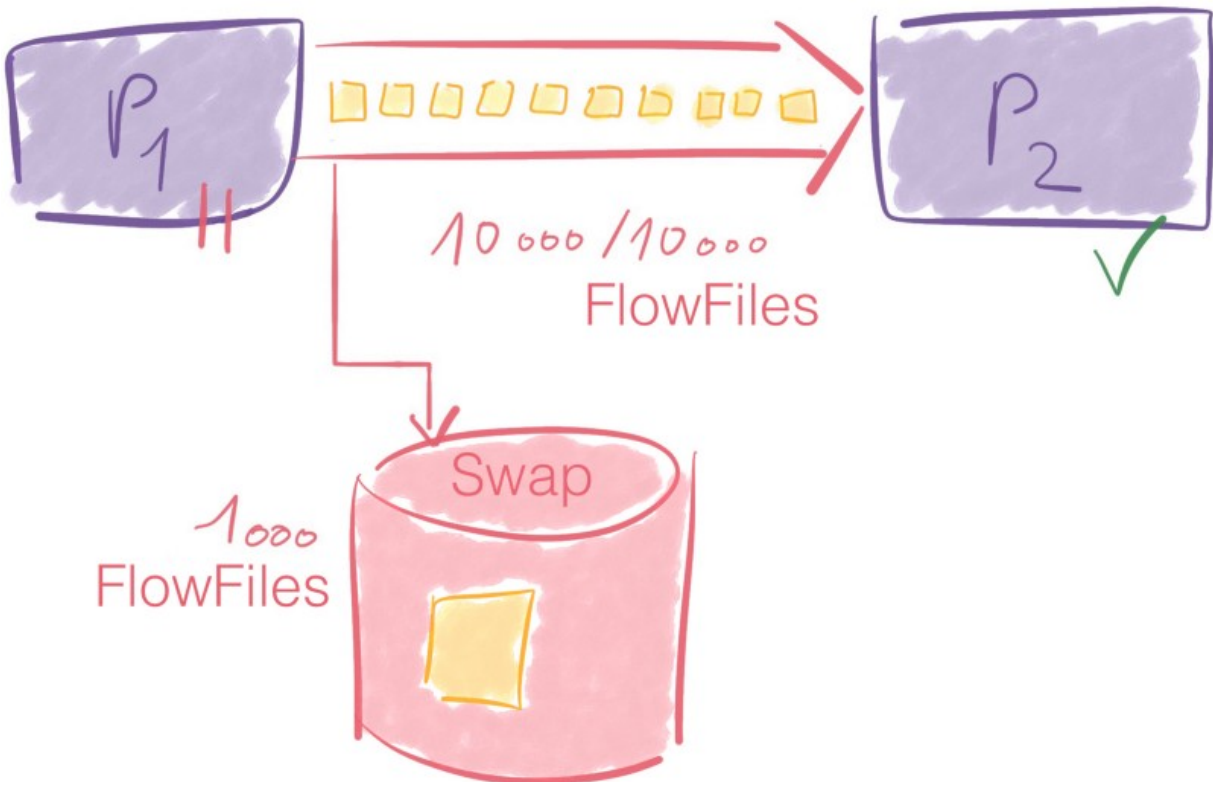


Figure 8 - Nifi Backpressure

4.4 Kubernetes

4.4.1 Container Orchestration

In such Microservices architecture, applications are so further broken up into various discrete services that are each packaged in a separate container. The benefit, especially for organizations that adhere to continuous integration and continuous delivery (CI/CD) practices, is that containers are scalable and ephemeral—instances of applications or services, hosted in containers, come and go as demanded by need.

But scalability is an operational challenge.

If you have ten containers and four applications, it's not that difficult to manage the deployment and maintenance of your containers. If, on the other hand, you have 1,000 containers and 400 services, management gets much more complicated. When you're operating at scale, container orchestration - automating the deployment, management, scaling, networking, and availability of your containers - becomes essential.

Container orchestration is all about managing the lifecycles of containers, especially in large, dynamic environments. Software teams use container orchestration to control and automate many tasks:

- Provisioning and deployment of containers

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

- Redundancy and availability of containers
- Scaling up or down to spread application load evenly across host infrastructure in a cost-effective way
- Movement of containers from one host to another if there is a shortage of resources in a host, or if a host dies
- Allocation of resources between containers
- External exposure of services running in a container with the outside world
- Load balancing of service discovery between containers
- Health monitoring of containers and hosts
- Configuration of an application in relation to the containers running it
- Upgrade version of containers without any downtime

When using a container orchestration tool, like Kubernetes, you typically describe the configuration of your application in a YAML or JSON file, depending on the orchestration tool. These configurations files are where you tell the orchestration tool where to gather container images, how to establish networking between containers, how to mount storage volumes, and where to store logs for that container.

Typically, teams will branch and control versions of these configuration files so they can deploy the same applications across different development and testing environments before deploying them to production clusters.

Containers are deployed onto hosts, usually in replicated groups. When it's time to deploy a new container into a cluster, the container orchestration tool schedules the deployment and looks for the most appropriate host to place the container based on predefined constraints (for example, CPU or memory availability). You can even place containers according to labels or metadata or according to their proximity in relation to other hosts - all kinds of constraints can be used.

Once the container is running on the host, the orchestration tool manages its lifecycle according to the specifications you laid out in the container's definition file (for example, its Dockerfile).

4.4.2 Kubernetes

Originally developed by Google as an offshoot of its Borg project, Kubernetes has established itself as the de facto standard for container orchestration. It's the flagship project of the Cloud Native Computing Foundation, which is backed by such key players as Google, Amazon Web Services (AWS), Microsoft, IBM, Intel, Cisco, and RedHat.

Kubernetes continues to gain popularity with DevOps practitioners because it allows them to deliver a self-service Platform-as-a-Service (PaaS) that creates a hardware layer abstraction for development teams. Kubernetes is also extremely portable. It runs on Amazon Web Services (AWS), Microsoft Azure, the Google Cloud Platform (GCP), or in on-premise installations. You can move workloads without having to redesign your applications or completely rethink your infrastructure—which helps you to standardize on a platform and avoid vendor lock-in.

Kubernetes is undoubtedly the most hyped orchestration tool, and also the most feature-filled orchestration tool available. But this could be one of its major disadvantages: it often offers more features than actually needed and then adds unneeded complexity.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

4.4.3 Kubernetes Components

- **Cluster:** A **cluster** is a set of **nodes** with at least one **master node** and several **worker nodes** that can be virtual or physical machines.
- **Node:** A **node** is a container runtime engine.
- **Kubernetes master:** The **master** manages the scheduling and deployment of application instances across **nodes**, and the full set of **services** the **master** node runs is known as the control plane. The **master** communicates with nodes through the Kubernetes API server. The scheduler assigns nodes to pods (one or more containers) depending on the resource and policy constraints defined.
- **Kubelet:** Each Kubernetes **node** runs an agent process called a **kubelet** that's responsible for managing the state of the **node**: starting, stopping, and maintaining application containers based on instructions from the control plane. A **kubelet** receives all of its information from the Kubernetes API server.
- **Pods:** The basic scheduling unit, which consists of one or more containers guaranteed to be co-located on the host machine and able to share resources. Each **pod** is assigned a unique IP address within the **cluster**, allowing the application to use ports without conflict. You describe the desired state of the containers in a **pod** through a YAML or JSON object called a PodSpec. These objects are passed to the **kubelet** through the API server.
- **Deployments, replicas, and ReplicaSets:** A **deployment** is a YAML object that defines the **pods** and the number of container instances, called **replicas**, for each **pod**. You define the number of **replicas** you want to have running in the **cluster** via a **ReplicaSet**, which is part of the **deployment** object. So, for example, if a **node** running a **pod** dies, the **replica set** will ensure that another **pod** is scheduled on another available **node**.

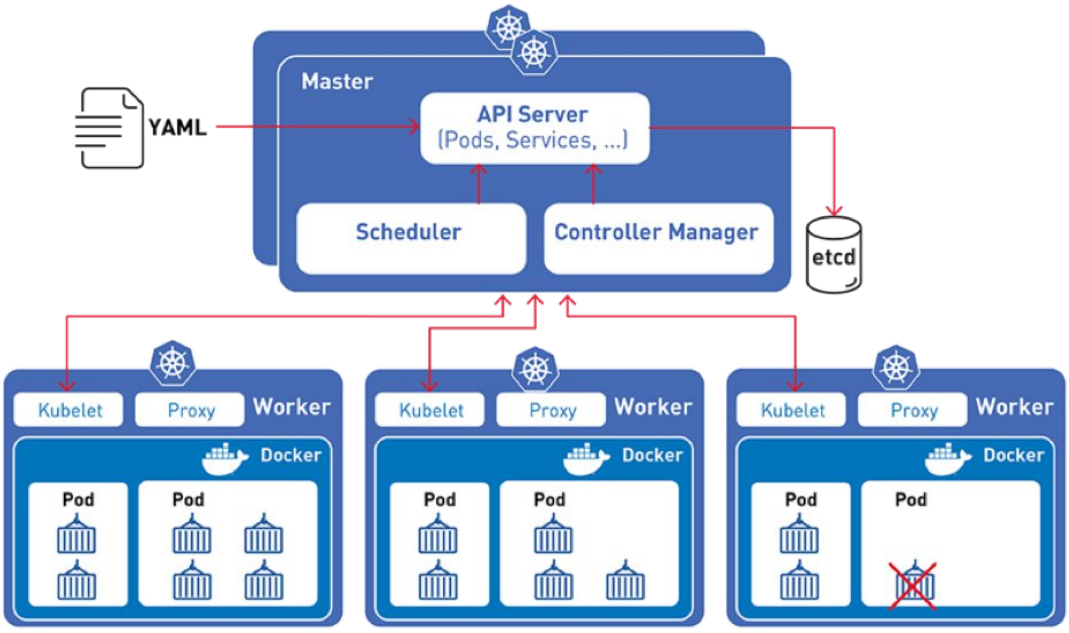


Figure 9 - Kubernetes Architecture

- **Service:** A **service** is an abstract way to expose an application running on a set of **pods** as a network service.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

- **kube-proxy: Kube-proxy** is a network proxy that runs on each node in the cluster, implementing part of the Kubernetes **Service** concept. **kube-proxy** maintains network rules on **nodes**. These network rules allow network communication to the **pods** from network sessions inside or outside of the **cluster**. **kube-proxy** uses the operating system packet filtering layer if there is one and it's available. Otherwise, **kube-proxy** forwards the traffic itself.
- **Ingress: Ingress** is an API object that manages external access to the **services** in a **cluster**, typically HTTP.

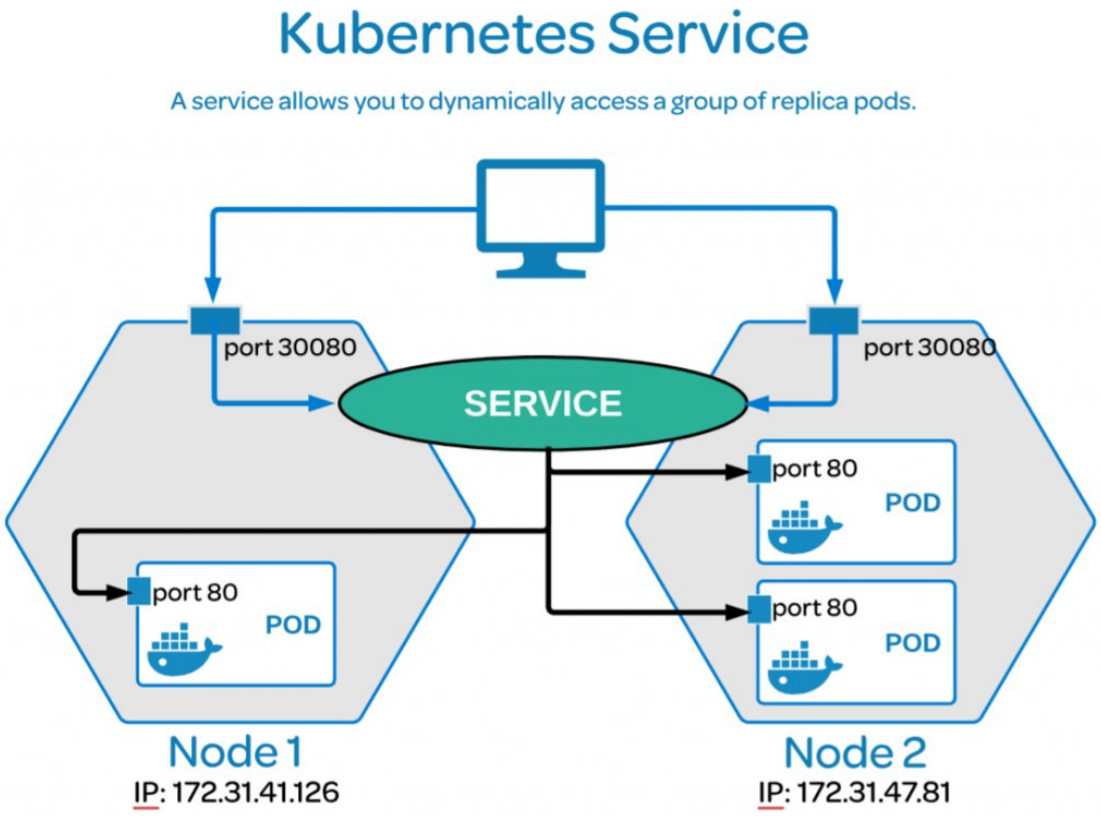


Figure 10 - Kubernetes Service

4.5 Data Fusion Bus (DFB)

Data Fusion Bus enables organizations in developing, deploying, operating and managing a big data environment with an emphasis on real-time applications. It combines the features and capabilities of several big data applications and utilities within a single platform.

The key capabilities of DFB are:

- Real-time monitoring and event-processing, semantic fusion of events not coinciding in time.
- Data aggregation from heterogeneous data sources and data stores.
- Real-time analytics, offering ready to use Machine Learning algorithms for classification, clustering, regression, anomaly detection.
- An extendable and highly customizable Interface (REST API and Web app) for configuring analytics, manipulation and filtering. It also includes functionality for managing the platform.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

DFB will be used in ROXANNE for:

- Aligning data streams for time and granularity: Event (e.g. landmark reference, Name reference) - time merge of different Kafka streams
- Providing granularity by creating aggregations in live or historical datasets when required (min, max, average, distinct values, etc.)
- Providing ML support to WP6: off-line creation of ML models based on historical data (ii) real-time clustering and classification of events (iii) persistent storage of metadata (iv) and exporting real-time results to other services through Kafka streams.

The main building blocks of DFB to be used in ROXANNE are:

- Data Analytics: DFB supports batch processing and stream processing with Kafka Streams & KSQL, Apache Spark and Spark Streaming.
- DFB Core: responsible for providing business logic to DFB and managing all the data flows. DFB Core exposes a REST API for configuring and running ML related tasks.

DFB relies on Kafka for streaming data input/output and Elasticsearch for persistent storage of data and metadata.

5. Modules Design – Services Interfaces

The architecture and frameworks described below allow each ROXANNE technical partner to work very efficiently. The components only need to agree on REST interface. As long as this interface is defined, a technical partner could work on is own, using any programming language or libraries. The final container is self sufficient and will not have any impact on the others while running.

In the following subsections, we present the first version of interfaces for the components which will be integrated into the ROXANNE platform. We also include the points which are still under discussion as of writing of this deliverable as footnotes in the respective subsections. The final version of these descriptions will be available in D5.3 Final set of speech/text/video technologies, to be delivered in M33.

Following components will be chained in one or several processing chains to produce knowledge from raw data (audio, text).

Some of these components needs to be trained on data to deliver optimal results. Such training features, based on LEA real data, will be described later in D7.4 Description of ROXANNE platform V1, to be delivered in M20.

The status of each components, the global system health will also be described later in D7.4 Description of ROXANNE platform V1, to be delivered in M20.

5.1 Speaker Diarization, Speech Activity Detection and Code-Switching Detection

The goal of these services is to pre-process the audio data, and to segment the audio into pieces which would ideally contain a homogeneous stretch of acoustic conditions. In particular,

Speech Activity Detection (SAD) determines the parts of audio which contain only speech and marks their endpoints, removing or labeling the non-speeech parts;

Speaker Diarization (SD) chops the speech content into pieces in which only voice of a single speaker is present. This service also includes Speaker Identification, the process of determining the identity of a speaker and/or their personal characteristics, such as age and gender;

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Code-Switching Detection (CSD) aims to find the different languages uttered in the spoken content, marking the times where a language-shift appears (e.g., a German speaker using English words in-between his/her sentence).

Input:

- Job Id : UUID
- Path to audio/wav file[1]

Output:

- JSON structure containing
- task_id: UUID
- task_type : sd (speaker diarization), sad (speech activity detection), or csd (code-switching detection)

- A list of segment clusters:
    - A label assigned to a cluster: a speaker identifier for SD[2], a tag indicating the gender of the speaker, a tag indicating the age of the speaker, a tag chosen from “speech”, “noise” and “silence” for SAD and an ISO639-3 formatted language for CSD.
    - A list of the segments in the cluster:
        - Start offset in the source wav file in milliseconds
        - End offset in the source wav file in milliseconds

Example :

Input :

```
[
  {
    "job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
    "path": file:///home/doc/audio.wav
  }
]
```

[1] The native audio format of the ROXANNE platform is defined to be the waveform audio file format (.wav). Support for other audio formats, codecs and containers (MP3, OGG, AMR, 3GP, GSM, etc.) will be considered in the future. The “audio/wav” expression here refers to the MIME type as is used in web development.

Depending on the actual technology chosen for shared storage, the term “path” may refer to unique identifiers or actual paths.

[2] Ideally, the unique identifier here would be a global one, such that for instance, if the same speaker is recognized in another audio file it would be possible link it to the same identity, rather than giving that person a new identifier. The label (presumably for visualization) would then be one of the possible names of that person, who might also be known under different aliases. All of these aliases should be stored in a central place for the speaker.

The speaker diarization component will also contain the detected age range and gender of the speaker. The categorization of the age and gender information is still to be discussed. An approach would be to define age ranges in decades ([0-10],[11-20],[21-30],...), another one is to use broad categories for both age and gender such as (Male-Adult / Female-Adult / Child). In either approach, we will consider the cases where the recognition result of the analysis component opposes the information recorded in the central repository.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

Output :


```
[
  {
    "task_id": "30379fcd-273d-47a9-ad37-278f56a55b62",
    "path": "file:///home/doc/audio.wav"
    "task_type": "sd",
    "entities": [
      {
        "label": "Spk1",
        "uuid": "40fd5f77-5564-44e1-9aee-4d0dd3723ee1",
        "segments": [
          {
            "start": "9410",
            "end": "32380",
            "confidence": "0.7"
          },
          {
            "start": "229720",
            "end": "244680",
          }
        ]
      },
      {
        "label": "Spk2",
        "uuid": "40fd5f77-5564-44e1-9aee-4d0dd3723ee2",
        "segments": [
          {
            "start": "139340",
            "end": "211960",
          }
        ]
      }
    ]
  }
]
```


5.2 Speaker Comparison/clustering

The difference between this module and the previouCreatingCreatingCreatingCreatingCreatingCreatingCreatingCreatingCreatingCreatings is that this module takes as input several recordings (or their voiceprints) instead of one. The recordings can be compared in several different ways. In the “speaker verification” scenario, one recording (the “test” recording) is compared against one or more recordings from the same speaker (the “enroll” recordings) and the system is supposed to tell how likely it is that the speaker in the test recording is the same as the speaker in the enroll recordings. In the “speaker identification” scenario, a test recording is compared to several enroll recordings and the system is supposed to tell which of enroll recordings is

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

most likely to be from the same speaker as the test recording. In the “speaker clustering” scenario, the system is supposed to find the most likely grouping of the utterances into speakers, i.e., find which recordings come from the same speaker.

Input:

- Job Id : UUID
- Files + auxiliary information, e.g., from the module described in Section 5.1
- Task description

Output

- task_id: UUID
- Likelihoods (or posteriors)

5.3 Spoken Language Recognition

This service allows to automatically recognize the language from speech data.

Input :

- Job Id : UUID
- path to audio/wav file
- (Optional) segments
    - start of the audio segment (in milliseconds)
    - end of the audio segment (in milliseconds)

Output :

- Task id in UTF-8
- Type of the task: audio-lid
- Language in ISO639-3[3]
- Confidence score [0..1]
- (Optional) segments

[3] The ISO639-3 statement here (and anywhere else in the document where a language identifier is concerned) is only a placeholder to represent a common (official) code table, not a final decision. The ISO639-3 code table does not differentiate between regional variations of some languages, including English (British/American/Indian, etc.). Therefore for practical purposes of ROXANNE, these language codes may be extended with a country code. Another idea in this respect is to include additional features such as "non-native vs native" through an accent detection component and the central speaker information repository, to allow for definitions like "native French speaker speaking English".

Independently of how the language identifier is being set, this component may also output multiple (possible) languages with respective confidence scores rather than a single one; similar to an ASR N-best list.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

- start of the audio segment (in milliseconds)
- end of the audio segment (in milliseconds)
- language in ISO639-3
- confidence score [0..1]

Example :

Input :

```
[
 {
  "job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
  "path": "file:///home/doc/audio.wav",
  "segments": [
   { "start": "0", "end": "323850" },
   { "start": "33800", "end": "65000" }
  ]
 }
]
```

Output :

```
 [
   {
     "task_id": "30379fcd-273d-47a9-ad37-278f56a55b62",
     "task_type": "audio-lid",
     "language": "fre",
     "global_confidence": "0.6",
     "segments": [
       {
         "start": "0",
         "end": "32380",
         "language": "fre",
         "confidence": "0.9"
       },
       {
         "start": "33800",
         "end": "65000",
         "language": "eng",
         "confidence": "0.7"
       }
     ]
   }
 ]
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

5.4 Automatic Speech Recognition (ASR)

This service allows to automatically transcribe the speech recording, i.e., convert the spoken language into a sequence of words. We aim at a multilingual setting, allowing different languages of interest to be deployed.

Input :

- Job Id : UUID
- path to audio/wav file
- An optional list of segments for pre-segmented audio with the segments correspond to speech sections which are to be transcribed. If this segmentation information is not provided, the whole input file will be transcribed.
- language in ISO639-3
- Number of desired N-best hypotheses (default 1, may not be supported by all ASR systems)

Output :

- Task id in UTF-8
- Type of the task: speech recognition (asr)
- A list of segments (if there is only one segment, it should span the whole file)
    - Vendor-specific segment-dependent metadata, such as
        - Characteristics and type of audio
        - Estimated noise level

  The following items pertain only to segments containing speech
    - Language of the segment
    - N-best hypotheses
        - Recognized word
            - Start offset in the source file in milliseconds
            - End offset in the source file in milliseconds
            - Confidence score of the word
        - N-best score
- transcription: text of the recognized speech (the transcription of the individual segments will be connected by full stops only)
- Global confidence score [0..1]

Example :

Input :

```
[
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
  {
    "job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",,
    "path": "file:///home/doc/audio.wav "
     "segments": [
      {
        "src_language": "eng",
        "start": "9410",
        "end": "32380",
      },
      {
        "src_language": "eng",
        "start": "229720",
        "end": "244680",
      }
    ]
  }
]
```

Output :

```
[
  {
    "task_id": "30379fcd-273d-47a9-ad37-278f56a55b62",
    "task_type": "asr",
    "segments ": [
      {
        "metadata": {
             "non_speech_ratio": "0.1539" // just an example
             "type": "speech"
      },
      "nbests": [
           {  // hypothesis1
               "words": [
                    {
                        "text": "awesome",
                        "start": "9410",
                        "end": "20545",
                        "confidence": "0.89"
                    },
                    {
                        "text": "speech",
                        "start": "20900",
                        "end": "32380",
                        "confidence": "0.95"
                    }
                ]
                "score": "0.9"
           }  // end of hypothesis1
    ]
   "transcription": "Awesome speech.",  //natural casing and full stop at the end
   "global_confidence": "0.9",
  }
]
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

5.5 Named Entity Recognition and Relation Extraction

This service allows to automatically recognise named entities (person name, location etc.) in the text. After named entities are detected, we further look for relations between these entities (e.g. <Obama, born in, Hawaii>). Both recognized entities and relations will be returned to the end-users.

Input :

- job_id: UUID
- Source language ISO 639-3
- Text content in UTF-8
- set of labels: set of NE labels. This is an optional input as a default will be given.
- model_identifier: UUID for a trained NN model used for NER. This is an optional input as a default will be given.

Output :

- task_id: UUID
- task_type : ner
- Entities
    - type of entities
    - entity name[4]
    - uuid of entity
    - list of segments :
        - startoffset,
        - endoffset,
        - confidence
- Relations (Optional, if not present, the service performs named entity detection only)
    - type of relation
    - uuid of relation
    - entity start uuid
    - entity end uuid
    - list of segments :
        - startoffset,
        - endoffset,
        - confidence

Example :

Input :

```
[
```

[4] This is a lemmatized version of the entity name with the full surface form specified in the list of segments. Different spellings/cross language/etc. would require an entity disambiguation step which is not part of this specific service.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
  {
    "job_id":"ee76a60a-b32c-4151-810e-c21817e3d142",
    "src_language":"fre",
    "content":"A Washington, Mr Bush a déclaré 'Ici à Washington, nous ..."
  }
]
```

Output :

```
[
  {
    "task_id": "30379fcd-273d-47a9-ad37-278f56a55b62",
    "task_type": "ner",
    "global_confidence": "0.7",
    "entities": [
      {
        "token": "Washington",
        "uuid": "40fd5f77-5564-44e1-9aee-4d0dd3723ee1",
        "type": "LOCATION",
        "segments": [
          {
            "start": "2",
            "end": "12",
            "confidence": "0.5"
          },
          {
            "start": "39",
            "end": "49",
            "confidence": "0.7"
          }
        ]
      },
      {
        "label": "Bush",
        "candidate": "false",
        "uuid": "40fd5f77-5564-44e1-9aee-4d0dd3723ee2",
        "type": "PERSON",
        "segments": [
          {
            "start": "17",
            "end": "21",
            "confidence": "0.7"
          }
        ]
      }
    ],
    "relations": [
      {
        "uuid_rel": "40fd5f77-5564-44e1-9aee-4d0dd3723ee3",
        "uuid_src": "40fd5f77-5564-44e1-9aee-4d0dd3723ee1",
        "uuid_tgt": "40fd5f77-5564-44e1-9aee-4d0dd3723ee2",
        "type": "PERSON_LOCATION",
        "segments": [
          {
            "start": "17",
            "end": "35",
            "confidence": "0.3"
          }
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
          ]
        }
      ]
    }
]
```

5.6 Text Language Identification (LID)

This service allows to automatically identify the language used in the text. Note this task is optional (not mentioned in the grant agreement). However, it could be delivered if it benefits the target users.

Input :

- job_id: UUID
- Text content in UTF-8
- Number of desired N-best hypotheses (default 1, may not be supported by all LID services)

Output :

- task_id: UUID
- N-best
    - language: ISO 639-3
    - score

Example :

Input :

```
[
  {
    "job_id":"ee76a60a-b32c-4151-810e-c21817e3d142",
    "content":"A Washington, Mr Bush a déclaré 'Ici à Washington, nous ..."
  }
]
```

Output :

```
[
  {
    "task_id": "30379fcd-273d-47a9-ad37-278f56a55b62",
    "nbests": [
   {
      "language": "fre",
      "score": "0.99"
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
  },
  {
      "language": "cat",
      "score": "0.01"
  }
  ]
 }
]
```

5.7 Sentiment Analysis (SA)

This service allows to automatically determine the polarity of sentences (positive/negative/mixed/neutral) in a textual input (either transcriptions of a recording or plain text). Note this task is optional (not mentioned in the grant agreement). However, it could be delivered if it benefits the target users.

Input :

- job_id: UUID
- src_language: Source language ISO 639-3
- Text content in UTF-8

Output :

- task_id: UUID
- response: The overall score for the whole input
    - features: Polarity values and decision
- sentences: The scores for individual sentences
    - content: Sentence input
    - features: Polarity values and decision

Input :

```
[
  {
    "job_id":"ee76a60a-b32c-4151-810e-c21817e3d142",
    "src_language":"fre",
    "content":"A Washington, Mr Bush a déclaré 'Ici à Washington, nous ..."
  }
]
```

Output :

```
[
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
  {
    "task_id": "30379fcd-273d-47a9-ad46-278f56a55b62",
    "response": {
       "features" {
           "positiveSentiment": 7
           "negativeSentiment": 3,
           "polarity": "POSITIVE"
       }
   },
   "sentences": [
       {
           "content": "A Washington, Mr Bush a déclaré",
           "features": {
               "positiveSentiment": 1
               "negativeSentiment": 1,
               "polarity": "MIXED"
           }
       }
   ]
  }
]
```


5.8 Place Diarization

The objective of the place diarization service is (1) to identify video segments related to the same place and (2) linking videos observing the same place, through image content indexing and similarity evaluation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

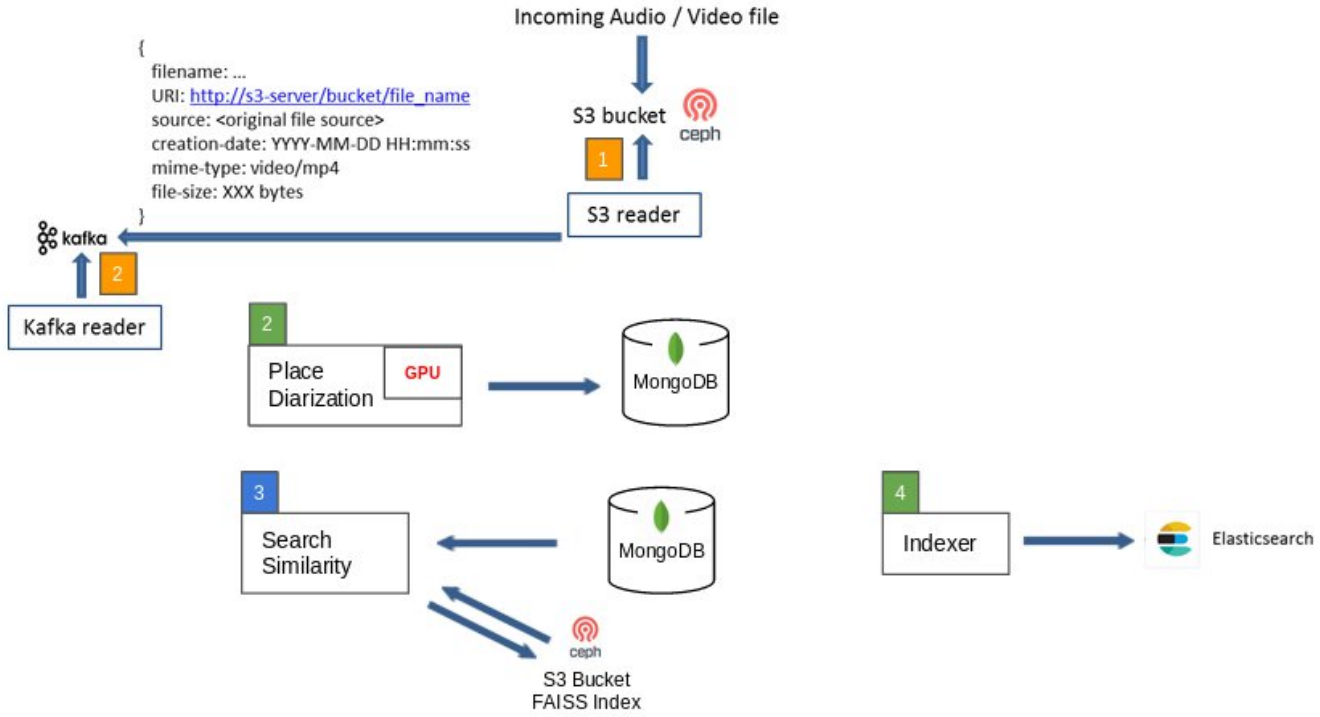


Figure 11 - Place diarization processing chain

The Place Diarization architecture includes:

- 3 separate modules:
- 2 event listeners : S3 reader and Kafka reader
- 1 kafka topic

Kafka topic (stream of records): A topic is a category or feed name to which records are published, similar to a message queue. Topic in kafka is always multi-subscriber; that is, a topic can have zero, one or many consumers that subscribe to the data written to it.

In this case:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

- The publisher is the S3 Reader, which publishes messages containing the path of the video uploaded to s3 and metadata. Each published message corresponds to a video uploaded to the s3 bucket. S3 Reader is executed each time a new object (video) is uploaded to the s3 bucket.

- The subscriber is the Kafka Reader (consumer), which has the role of subscribing to a kafka topic and retrieving new messages. This module must periodically (either after a defined time or after receiving a defined number of messages) call another module via a Rest API. This module receives as input the messages from the Kafka topic and returns these messages to a REST API.

Three modules must run in a Docker:

- Place diarization: This module requires a GPU. It receives as input the messages coming from the kafka reader (consumer) and analyzes, extracts signatures, performs a place diarization (i.e identification of temporal segments in a given video related to the same location) of each video and pushes the results (as JSON objects) to MongoDB.

-Similarity Search: A periodic task. This module retrieves new records in MongoDB, synchronizes the FAISS index (index stored on a bucket s3) and run similarity search between the new records and the whole database. This module returns results (location matching between several videos) in a JSON object.

-Indexer: Takes as input a JSON from the Similarity Search module and inserts it into an ElasticSearch database for further visualization / query purposes.

The "Place Diarization Service" must run in a docker with a GPU (rtx 2080 ti for example).

Example document mongoDB:

- Job_id : uui
- status instance : new / processed
- Video_id : uui
- task_id : uui
- list of places :
    - id_place
    - segments:
        - start
        - end
    - signatures

Place diarization

Input :

- job_id : UUI
- video_path : path s3 file

Output :

- Job_id : uui
- Video_id : uui
- task_id : uui
- list of places :
- id_place
    - confidence
    - segments:
        - start

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

- end
- signatures

Example :

Input :


```
[
{
"job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
"video_path": "s3://bucket/video/1365156.mp4",
}
]
```


Output :


```
[
{
"job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
"task_id": "fds1ds56f-sdfg-sd5f-z8e9-qs561dsq6fs",
"video_id": "79071b23-83f1-4e61-92cb-f2c41750c538",
"places": [
{
"place_id": "97853951-7041-410f-bf3e-89739ad1a2b1",
 "confidence": "0.86"
"segments": [
{
"start": 0,
"end": 3550
},
{
"start": 9826,
"end": 12026
}
],
"signatures" : [{"1582238376":[8.85133922e-01, 1.08143437e+00, 1.20291400e+00,
-1.15219557e+00, -4.11096662e-01, 1.15465784e+00,
```

3.78168017e-01, 8.45648706e-01]},]

},

{

"place_id": "f2d3fd4f-dd2c-47ce-814c-e6a4cb071ea5",

"confidence": "0.98"

"segments": [

{

"start": 26532,

"end": 28210

},

{

"start": 89513,

"end": 92513

}

],

"signatures" : [{"15825317489":[9.5616e-01, 1.54654654e+00, 3.456540e+00,

-1.15219557e+00, -3.11096662e-01, 1.15465784e+00,

3.78168017e-01, 1.45648706e-01]},]

}

]

}

]

Search Similarity Service

Input:

- Job Id : UUI

Output:

- job_id
- task_id
- nb matching
- list of matching (relation):
  - id matching
  - confidence
  - list of matches:
    - id video
    - path to video

- id place

Example:
Input :


```
[
{
"job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
}
]
```


Output :


```
[
  {
    "job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
    "task_id": "bf0a1ee4-d88f-436b-b4a1-149adb7a9690",
    "nb_matching": 1,
    "matching": [
      "match_id": "8a171b24-407c-411b-b3a0-47c7bc758ad0"
      "confidence": "0.97"
      "matches": [
      {
        "video_id": "79071b23-83f1-4e61-92cb-f2c41750c538",
        "path_to_video": "s3://bucket/video/1.mp4",
        "place_id": "97853951-7041-410f-bf3e-89739ad1a2b1",
      },
      {
        "video_id": "8f5b30b7-486d-4ab3-b217-5246b2ce5d34",
        "path_to_video": "s3://bucket/video/4.mp4",
        "place_id": "dec2cc2a-67f5-45ae-a396-3668b647e55c",
      }
      ]
    ]
  }
]
```


5.9 Face Diarization

Description: The objective of the Face Diarization service is (1) to identify video segments related to the same persons/faces and (2) linking videos containing the same person (without identifying the person), through face signature indexing and similarity evaluation. The proposed architecture is identical to the Place Diarization.

Face Diarization Service

Input :

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

- job_id : UUI
- video_path : path s3 file

Output :

- Job_id : uui
- Video_id : uui
- task_id : uui
- list of person :
  - id_person
  - confidence
  - segments:
    - start
    - end
  - signatures

Example :

Input :


```
[
{
"job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
"video_path": "s3://bucket/video/1365156.mp4",
}
]
```


Output :


```
[
{
"job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
"task_id": "fds1ds56f-sdfg-sd5f-z8e9-qs561dsq6fs",
"video_id": "79071b23-83f1-4e61-92cb-f2c41750c538",
"persons ": [
{
"person_id": "97853951-7041-410f-bf3e-89739ad1a2b1",
 "confidence": "0.86"
"segments": [
{
"start": 0,
"end": 3550
},
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
{
"start": 9826,
"end": 12026
}
],
"signatures" : [{"1582238376":[8.85133922e-01, 1.08143437e+00, 1.20291400e+00,
-1.15219557e+00, -4.11096662e-01, 1.15465784e+00,
3.78168017e-01, 8.45648706e-01]},]
},
{
"person_id": "f2d3fd4f-dd2c-47ce-814c-e6a4cb071ea5",
 "confidence": "0.98"
"segments": [
{
"start": 26532,
"end": 28210
},
{
"start": 89513,
"end": 92513
}
],
"signatures" : [{"15825317489":[9.5616e-01, 1.54654654e+00, 3.456540e+00,
-1.15219557e+00, -3.11096662e-01, 1.15465784e+00,
3.78168017e-01, 1.45648706e-01]},]
}
]
}
]
```


Search Similarity Service

Input:

- Job Id : UUI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Output:

- job_id
- task_id
- nb matching
- list of matching (relation):
    - id matching
    - confidence
    - list of matches:
        - id video
        - path to video
        - id person

Example:
Input :

```
[
{
"job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
}
]
```

Output :

```
[
  {
    "job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
    "task_id": "bf0a1ee4-d88f-436b-b4a1-149adb7a9690",
    "nb_matching": 1,
    "matching": [
      "match_id": "8a171b24-407c-411b-b3a0-47c7bc758ad0"
      "confidence": "0.97"
      "matches": [
      {
        "video_id": "79071b23-83f1-4e61-92cb-f2c41750c538",
        "path_to_video": "s3://bucket/video/1.mp4",
        "person_id": "97853951-7041-410f-bf3e-89739ad1a2b1",
      },
      {
        "video_id": "8f5b30b7-486d-4ab3-b217-5246b2ce5d34",
        "path_to_video": "s3://bucket/video/4.mp4",
        "person_id": "dec2cc2a-67f5-45ae-a396-3668b647e55c",
      }
      ]
    ]
  }
]
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

5.10 Video2Text Service

Description: The objective of the Video2Text is to associate video segments with keywords coming from object detectors or semantic segmentation classes.

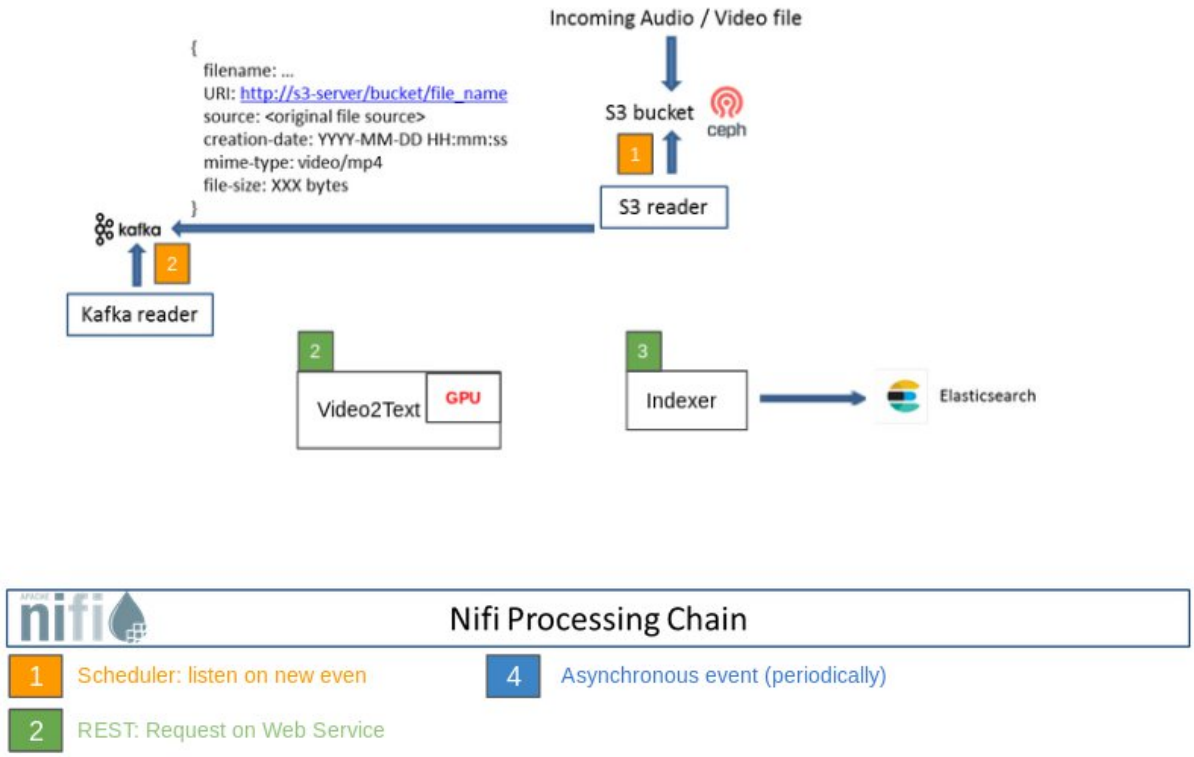


Figure 12 - Voice2Text processing chain

Input:

- Job Id : UUI

Output:

- job_id
- task_id
- list of segments :
  - id segment
  - start
  - end
  - list of keywords:
    - id object
    - label
    - confidence

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Example:
Input :

```
[ {
   "job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
 } ]
```

Output :

```
[
  {
    "job_id": "ee76a60a-b32c-4151-810e-c21817e3d142",
    "task_id": "bf0a1ee4-d88f-436b-b4a1-149adb7a9690",
    "segments ": [
      {
        “id_segment” : “sd89f465-iuo456 -915 -qspod651 -7uytjh132ga”,
        “start” : 0,
        “end”: 162,
        “keywords” : [
            {
              “id_class” : 1,
              “label”: “person”,
             “confidence” : 0.95
            },
           {
              “id_class” : 3,
              “label”: “building ”,
             “confidence” : 0.99
            },
            {
              “id_class” : 9,
              “label”: “car ”,
             “confidence” : 0.86
            }
        ]
      }
    ]
  }
]
```


This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

5.11 Social Influence Analysis / Importance Individual Detection (Network Analysis Service)

Description: This service allows to identify most influential/important individuals from the relations or interactions among them by computing for each entity one or more importance scores

Input :

- job_id: UUID
- Relations/ Interactions
  - source individual
    - individual id
    - individual properties
      - type, place, age, gender, spoken languages, etc.
      - recognition confidence
  - destination individual
    - individualid
    - individual properties
      - place, age, gender, spoken languages, etc.
      - recognition confidence
  - relation/interaction id
  - relation/interaction properties
    - type, associated content, etc.
    - recognition confidence

Output :

- job_id: UUID
- Individuals' importance score
  - entity id
  - importance score

Example :

Input :

```
{
        "job_id": "sia_xxxx",
        "relations":
                [
                        {
                                "id": "relation_xx0"
                                "source": {"id":"person_A", "type": "person"},
                                "destination":{"id":"person_B", "type": "person"}
                                "properties":{"type":"phone_call", "confidence": 0.9}
                        },
                        {
                                "id": "relation_xx1"
                                "source": {"id":"person_A", "type": "person"},
                                "destination":{"id":"person_C", "type": "organization"}
                                "properties":{"type":"work_for", "confidence": 0.9}
                        },
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
                ]

}
```

Output :

```
{
        "job_id": "sia_xxxx",
        "scores":[{"id": "person_A", "score": 0.3}, {"id": "person_B", "score": 0.2}, {"id": "org_C", "score": 0.1}]
}
```

5.12 Community Detection/ Network Clustering (Network Analysis Service)

Description: This service allows to identify cohesive groups of individuals from the observed relations or interactions among them

Input :

- job_id: UUID
- Relations/ Interactions
    - source individual
        - individual id
        - individual properties
            - place, age, gender, spoken languages, etc.
            - recognition confidence
    - destination individual
        - individual id
        - individual properties
            - place, age, gender, spoken languages, etc.
            - recognition confidence
    - relation/interaction id
    - relation/interaction properties
        - type, associated content, etc.
        - recognition confidence
- Number of communities (optional)

Output :

- task_id: UUID
- Communities
    - community id
    - community members
        - individual id
        - membership score

Example :

Input :

```
{
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
        "job_id": "scd_xxxx",
        "relations":
                [
                        {
                                "id": "relation_xx0"
                                "source": {"id":"person_A", "type": "person"},
                                "destination":{"id":"person_B", "type": "person"}
                                "properties":{"type":"phone_call", "confidence": 0.9}
                        },
                        {
                                "id": "relation_xx1"
                                "source": {"id":"person_A", "type": "person"},
                                "destination":{"id":"person_C", "type": "organization"}
                                "properties":{"type":"work_for", "confidence": 0.9}
                        },

                ],
        "num_communties": 2


}
```


Output :


```
{
        "job_id": "scd_xxxx",
        "communities":
                [
                        {"id": "comm_1", "members": [{"id": "person_A", “score”: 1.0},{"id": "person_B", “score”: 0.5}]},
                        {"id": "comm_2", "members": [{"id": "person_A", “score”: 0.7},{"id": "org_C", “score”: 1.0}]}
                ]
}
```


5.13 Latent Link Prediction (Network Analysis Service)

Description: This service allows to predict, for each input individual, the potential latent (or hidden) links to other entities based on the relations or interactions among them.

Input :

- job_id: UUID
- Relations/ Interactions
  - source individual
    - individual id

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 833635.

        - individual properties
            - place, age, gender, spoken languages, etc.
            - recognition confidence
    - destination individual
        - individual id
        - individual properties
            - place, age, gender, spoken languages, etc.
            - recognition confidence
    - relation/interaction id
    - relation/interaction properties
        - type, associated content, etc.
        - recognition confidence
- Source individuals (optional)

Output :

- task_id: UUID
- Predicted links for source individuals
    - individual
    - predicted destination individuals
        - individual id
        - confidence score

Example :

Input :

```
{
        "job_id": "slp_xxxx",
        "relations":
                [
                        {
                                "id": "relation_xx0"
                                "source": {"id":"person_A", "type": "person"},
                                "destination":{"id":"person_B", "type": "person"}
                                "properties":{"type":"phone_call", "confidence": 0.9}
                        },
                        {
                                "id": "relation_xx1"
                                "source": {"id":"person_A", "type": "person"},
                                "destination":{"id":"org_C", "type": "organization"}
                                "properties":{"type":"work_for", "confidence": 0.9}
                        },

                ],
        "sources": ["person_A", "org_C"]
}
```

Output :

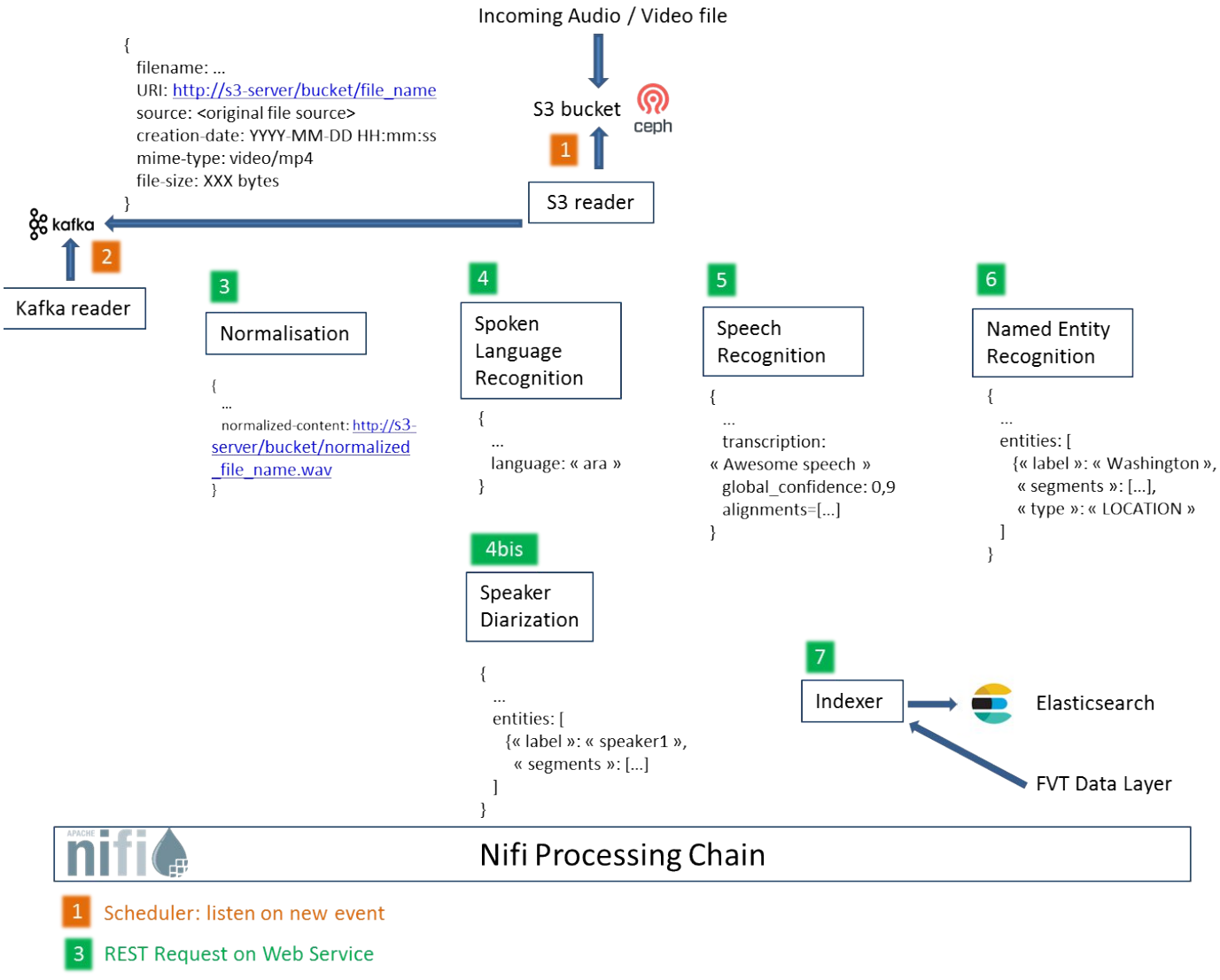
```
{
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
        "job_id": "slp_xxxx",
        "predtictions":
                [
                        {"source": "person_A", "destinations": [{"id": "person_D", "score": 0.2}]},
                        {"source": "org_C", "destinations": [{"id": "person_B", "score": 0.5}]}
                ]
}
```


5.14 Processing chain example

Here is described a potential processing chain running on ROXANNE platform, combining several Micro Services.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Figure 13 - Roxanne processing chain example

This Nifi processing chain is using 2 main components:

- Schedulers listening to new events (new file in S3 Object storage, new data in a Kafka topic...)
- HTTP Calls to REST Microservices

During the whole processing, the Nifi flow file is enriched by the result of the processing.

In the end, complete file and all processing results are stored In Elasticsearch indices for advanced search on data

FVT Data Layer is requesting periodically for newly processed data for generating insightful information that forms the basis of visualisation (see next paragraph).

6. Data visualisation framework description

The FVT (Forensics Visualization Tool) module is the user interface of ROXANNE platform. It provides the means to visualise social network relations and several indicators, insights derived from multi-modal data (i.e., text, audio and video). It enables the final user to explore data at a high level through several interconnected, interactive visualisations that also allow drilling into more detailed information to reveal hidden relations and insights. To achieve this goal, the toolkit follows an internal architecture that comprises of 3 layers, namely: data, business and presentation. This architecture provides the flexibility to adapt to various domains and data sources.

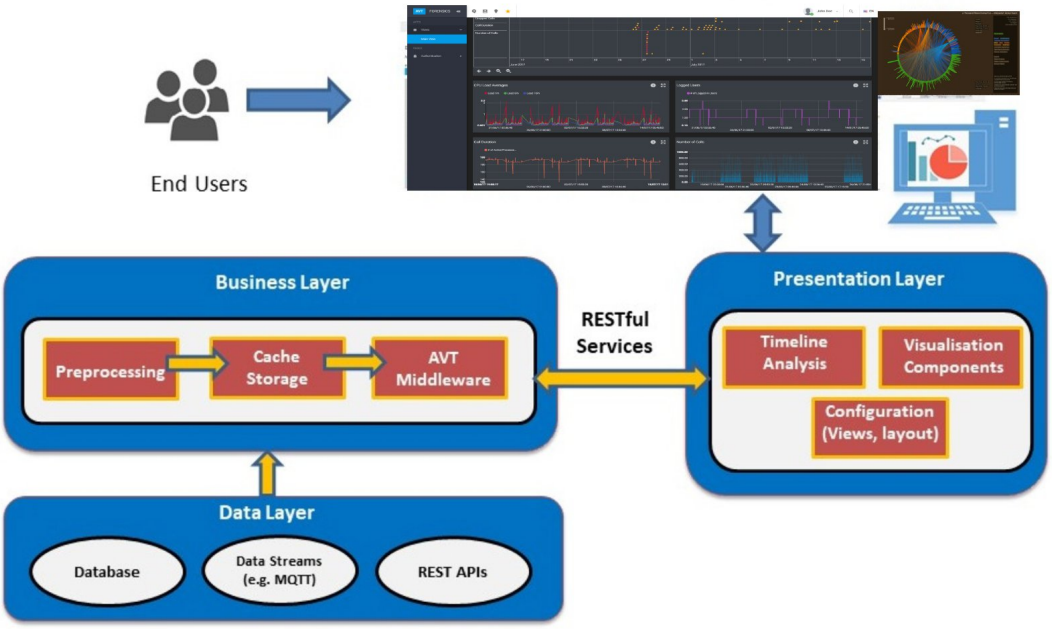


Figure 14 - The architecture of Forensics Visualization Tool

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

6.1 The Data layer

The Data layer includes the original data and the mechanisms that enable their retrieval by the business layer of the FVT. Data can come from several sources, including local databases, data streams (such as MQTT sources) and REST APIs. Data itself is domain-specific and in the case of ROXANNE can refer to outputs of speaker identification, speech recognition, as well as places, faces and other information extracted from audio and video.

6.2 The Business layer

The Business layer of FVT forms the backend system that is responsible for transforming input data to a format suitable for the visualisations, caching data for improved performance and serving the REST services required by the presentation layer. Each component is further described below.

1. Pre-processing

The Pre-processing component can be configured to retrieve data from one, or more, sources at user-defined intervals or triggered by end-users on demand. Then it applies required filtering and processing in order to extract information and compute statistics and indicators. The output is in JSON format, suitable for reading and further processing by the cache Creatingstorage component (see next).

2. Cache Storage

All pre-processed data is retrieved from the Event Pre-processing subsystem and stored in a Cache for quick retrieval by the Visualization Components. Since the historical data for each indicator do not change, related data is stored in a flat, reusable structure, containing only the timestamp and the relevant values. Additional information, such as communication payload, can be stored separately.

3. AVT Middleware

The AVT Middleware component handles the communication between the Cache and the Visualisation Tools. The component exposes a set of REST services that allow retrieval, sorting and filtering of event information by the Visualisation Tools.

6.3 The Presentation layer

The presentation layer includes several visualization elements, a timeline analysis component and the configuration files.

1. The visualizations component

This component is responsible for displaying diverse data types in an intuitive and interactive way. For this purpose, several visualisations formats are supported including line graphs, point graphs, bar charts, pie charts, grid/tables, scatterplots, bubble charts, chord diagrams, span charts, gauges, as well as selected combinations.

While bar charts and pie charts are some of the standard ways of data representation, we expect that chord diagrams will be very useful for displaying the inter-relations between suspects and identifying criminals' structure. More specifically, individuals would be arranged radially around a circle and interactions are drawn as arcs connecting them. The width of those arcs could depend on the frequency and/or duration of interactions. By selecting a certain individual, the operator can see personal details (if available) and filter interactions. This way, end users can get situational awareness of the system (where original data come from) at any given time and identify patterns.

2. The timeline analysis component

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Another important visualization element are line charts, especially when combined with the timeline analysis component that offers the functionalities for temporal analysis of indicator values. Suppose that an event, such as a suspicious phone call, occurs at time t0. Through the timeline control, the end-user may move back in time and see details about past, or future, related events and compare two different time points, in order to get insights of the data. Changes of the time window propagate to the rest of the visualisations so that they always follow the selected time window. The operator can configure the set of visualizations to be displayed according to its custom needs and preferences.

3. Configuration

The Configuration component includes a set of mechanisms and configuration files that ensure proper fit of the FVT to a case under investigation. Indicative parameters that can be configured include the visualisations shown on the dashboard network by default, alerts if certain rules are met, etc.

6.4 FVT integration regarding the architecture refinement

The visualization framework will be built based on the input received by other modules of the ROXANNE framework. As described earlier, this communication is foreseen to take place via a service-oriented approach, where each component exposes its outputs via web services that can be afterwards consumed by the visualization component. Dashboard elements, extracted analytics and alerts will be the main information to visualise at different levels of granularity that can reach the log files where the initial event was caught in.

The selected architectural style is REST and during the development period, we will continuously structure the service signatures that are going to be available so as to support the visualisation needs of the various components. It must be noted that since the project is still at an early stage, services are subject to define and additions of services are expected as development advances and user needs evolve.

7. Hardware requirements

Following table describes ROXANNE platform Hardware configuration.

Item	Reference	Quantity
Mother card	90SB0460-M0EAY0 ASUS Z10PE-D8 WS	1
CPU	Intel Xeon E5-2630V4 - 2.2 GHz - 10 cores	1
HDD	Seagate Barracuda ST6000DM003 - DD - 6 To	1
HDD	Seagate Barracuda ZA1000CM1A002 - SSD - 1 To	1
RAM	32Go DDR4 PC21300/2666Mhz 2RX8 CL19 1.2V 288 pins	8
GPU	ASUS DUAL-RTX2080TI-11G - GF RTX 2080 Ti 11 Go GDDR6 PCIe 3.0 x16 HDMI. 3 x	2

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

	DisplayPort. USB-C	

8. Conclusion

The purpose of this document was to describe the architecture that will be used for ROXANNE platform, a novel platform combining advances of speech, language and video technologies and criminal network analysis for supporting investigators in their daily work.

In addition to the high level architecture, it clearly defines the specification of each component and the technical means with which it will be integrated into the demonstrator.This specification is now used by the partners to supply their components for a first integration in order to prepare the first field test.

9. References

How Apache Nifi works : https://www.freecodecamp.org/news/nifi-surf-on-your-dataflow-4f3343c50aa2/

Apache Kafka documentation: https://kafka.apache.org/

Kubernetes documentation: https://kubernetes.io/

Docker documentation: https://www.docker.com/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

10. Appendix

Phonexia Speech Engine (VAD - voice activity detection)

https://download.phonexia.com/docs/spe/

Input:

curl, php, js

```
var options = {
  method: 'GET',
  uri_path: '/technologies/diarization',
  uri_params: {
    path: '/julia.wav',
    model: 'S'
  },
  callback: function(result)
  {
    console.log(result); // work with result
  }
};
async_request(options);
```

Output:

json / xml

Example

```
{
  "result": {
    "version": 3,
    "name": "DiarizationResult",
    "file": "julia_1.wav",
    "model": "S",
    "max_speakers": 2,
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
    "total_speakers": 2,
    "per_channel": true,
    "time_range": {
      "from_time": 2.5,
      "to_time": 5.5
    },
    "segmentation": [
      {
        "channel_id": 0,
        "score": 0,
        "confidence": 0,
        "start": 0,
        "end": 6700000,
        "word": "silence"
      },
      {
        "channel_id": 0,
        "score": 0,
        "confidence": 0,
        "start": 6700000,
        "end": 10200000,
        "word": "1"
      },
      ...
    ]
  }
}
```


segmentation[]	list of segments
segmentation[]->channel_id	channel index
segmentation[]->score	not used, always 0

segmentation[]->confidence	not used, always 0
segmentation[]->start	start time of word (HTK time units - 100ns)
segmentation[]->end	end time of word (HTK time units - 100ns)
segmentation[]->word	contains number of speaker or `silence`

Phonexia Speech Engine (LID - language identification)

https://download.phonexia.com/docs/spe/

Input:

curl, php, js

```
var options = {
  method: 'GET',
  uri_path: '/technologies/languageid',
  uri_params: {
    path: '/julia.wav',
    model: 'L',
    language_pack: 'default'
  },
  callback: function(result)
  {
    console.log(result); // work with result
  }
};

async_request(options);
```

Output:

json / xml

Example

```
{
  "result": {
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
    "version": 2,
    "name": "LanguageIdentificationResult",
    "file": "julia_1.wav",
    "language_pack": "default",
    "model": "L",
    "time_range": {
      "from_time": 2.5,
      "to_time": 5.5
    },
    "channel_scores": [
      {
        "channel": 0,
        "scores": [
          {
            "name": "Afan_Oromo",
            "score": -8.212487
          },
          {
            "name": "Albanian",
            "score": -7.554749
          },
          ...
        ]
      }
    ]
  }
}
```


Phonexia Speech Engine (SID4 - speaker identification)

https://download.phonexia.com/docs/spe/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

Input:

curl, php, js

```
var options = {
  method: 'GET',
  uri_path: '/technologies/speakerid4',
  uri_params: {
    path: '/julia.wav',
    model: 'L4',
    group: 'group1',
    audio_source_profile_1: 'test_profile_1',
    audio_source_profile_2: 'test_profile_2'
  },
  callback: function(result)
  {
    console.log(result); // work with result
  }
};
async_request(options);
```

Output:

json / xml

Example

```
{
  "result" : {
    "version" : 1,
    "name" : "SpeakerIdentification4MultiResult",
    "model" : "L4",
    "speaker_group" : "group1",
    "time_range": {
      "from_time": 2.5,
      "to_time": 5.5
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.

```
  },
  "results" : [
    {
      "file" : "julia_1.wav",
      "speaker_model" : "test1",
      "audio_source_profile_1" : "test_profile_1",
      "audio_source_profile_2" : "test_profile_2",
      "channel_scores" : [
        {
          "channel" : 0,
          "scores" : [
            {
              "score" : 15.753671
            }
          ]
        }
      ]
    },
    {
      "file" : "julia_1.wav",
      "speaker_model" : "test2",
      "audio_source_profile_1" : "test_profile_1",
      "audio_source_profile_2" : "test_profile_2",
      "channel_scores" : [
        {
          "channel" : 0,
          "scores" : [
            {
              "score" : -14.8087845
            }
          ]
        }
      ]
    }
```

```
    ]
  }
}
      },
      {
        "channel_id": 0,
        "score": 0,
        "confidence": 0,
        "start": 6700000,
        "end": 10200000,
        "word": "1"
      },
      ...
    ]
  }
}
```


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833635.