COVID X

COVID eXPONENTIAL PROGRAMME

GRANT AGREEMENT ID: 101016065

D2.3 – Final Sandbox Implementation and Services Provision

Revision: v1

Work Package	WP2
Due date	31/10/2021
Submission date	30/10/2021
Deliverable lead	INTRASOFT INTERNATIONAL
Version	1.0
Authors	Serafeim Tsironis, Filopoimin Lykokanellos, Themistoklis Anagnostopoulos, Sofia Tsekeridou (INTRASOFT) Despoina Gkatzioura (8Bells) Borja Aroyo, Silvia Uribe, Javier Serrano (UPM)
Reviewers	Rita Campos (F6S)

Abstract	This deliverable presents the final release of the COVID-X Sandbox and describes the services offered, including data ingestion, annotation, harmonization, visualization, federated learning and validation, interoperable data access and retrieval, API connectivity and management
Keywords	sandbox, data management, data description, data harmonization, data integration, visualization, data access, data search, data retrieval, federated learning, security

COVID-X has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101016065.

DOCUMENT REVISION HISTORY

Version	Date	Description of change	List of contributors
V0.1	07/09/2021	Initial ToC created	INTRA
V0.2	07/09/2021	ToC Review and Corrections	INTRA
V0.3	21/09/2021	Add Section 2, Section 3.2	INTRA
V0.4	05/10/2021	Add Section 3.3	8BELLS
V0.5	08/10/2021	Add Section 1, Update Section 3, Review and Corrections, Add Appendix – COVID-X Semantic Data Model	INTRA
V0.6	15/10/2021	Add Subsection 3.1.2	INTRA, 8BELLS
V0.7	18/10/2021	Update Subsection 3.1.2	8BELLS
V0.8	19/10/2021	Update Sections 2, 3.4	INTRA, UPM
V0.9	20/10/2021	Add Sections 3.1.1, 4, 5	INTRA
V0.10	22/10/2021	Updates in all sections, introduction and conclusions added	INTRA, 8BELLS
V0.11	26/10/2021	Final Corrections, Submission for internal review	INTRA
V1.0	30/10/2021	Final formatting	F6S

DISCLAIMER

COPYRIGHT NOTICE

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	**OTHER**	
Dissemination Level		
PU	Public, fully open, e.g., web	X
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential project and Commission Services	

EXECUTIVE SUMMARY

The current document presents the final software release of COVID-X Sandbox, presenting the services and all the technical details that related to these. In order to support functionality for seamless access to a set of healthcare data sources, COVID-X Sandbox is based on the ELK stack components to provide services for data processing, data harmonization, data indexing, data archiving, data querying, data visualization. On top of them, it implements the security mechanisms for unauthorized users to access personal data based on the well-established encryption protocols and services and on relevant technologies. COVID-X Sandbox is based on a highly modular architecture and all the components within this architecture realize and deliver one or more aggregated services. As the first release of COVID-X Sandbox was initially presented at D2.2 - First Sandbox implementation and services provision, this document focuses mostly on the updates regarding the system architecture, the offered services and any improved or new functionalities that became part of the Sandbox in the meanwhile. Furthermore, the document introduces the definition of unified data presentation model for the data coming for healthcare providers in order to support the COVID-X project use cases. Finally, an overview of the available COVID-X Sandbox deployment is presented along with a user guide that provide all the necessary instruction for accessing and using COVID-X Sandbox services.

TABLE OF CONTENTS

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

COVID-X	Innovation action supported by the European Commission in the framework of the EC call SC1-PHE-CORONAVIRUS-2020-2B
AI	Artificial Intelligence
Accelerate	Transfer technical, business, and ethical knowledge
OC	Open Calls
Single player	*See Single Solution*
Sigle Solution (SS)	Open call line for technology providers
Team Solution (TS)	Open call line for clinical partners working in a team with technology providers
FL	Federated Learning
CI/CD	Continuous Integration / Continuous Deployment
VCS	Version Control System
VPN	Virtual Private Network
ICD	International Classification of Diseases
FAIR	Findability, Accessibility, Interoperability, Reusability
ELK	Elasticsearch, Logstash, Kibana
CA	Certification Authority
PL	Programming Language
DSL	Domain Specific Language
SME	Small and Medium Enterprises
GDPR	General Data Protection Regulation
DPA	Data Protection Agreement
SSH	Secure Socket Shell

1 Introduction

1.1 Purpose of the Deliverable

This document presents and describes the final software release of COVID-X Sandbox with all the corresponding components and services offered in terms of COVID-X project. Following the work initially presented under D2.2 - First Sandbox implementation and services provision [2], the current document addresses all the new services and functionalities that have been added and all the improvements done to have a complete fully functional final version of COVID-X Sandbox.

In this documentation, we describe all the technical details and obstacles faced in terms of COVID-X Sandbox, compared to the initial version and since multiple use cases (as part of the Open Call 1 – OC1) became part of the entire COVID-X project in the meantime.

Furthermore, we describe all the aspects related to the deployment of both the local and the centralized deployments of COVID-X Sandbox, along with all the technical challenges that appeared and the suggested solutions to face them. In addition, we present a new functionality which will allow the possibility of permitting models training via Federated Learning.

Finally, we present an overview of the Unified Semantic Data Model which can be used as a template for all the data processed, stored and shared to external APIs, services and entities that need to access and retrieve information from the COVID-X Sandbox. The Unified Semantic Data Model is detailed in a specific section of D2.4 - First Report on interconnection of third parties, CI/CD and policies [3].

1.2 Structure of the Deliverable

The structure of this deliverable is the following:

- **Section 1** introduces the deliverable and explains its purpose and structure.
- **Section 2** describes all the updates and adjustments done regarding the COVID-X Sandbox CI/CD stack initially introduced in the first release.
- **Section 3** presents in more detail the implementation of the developed services and the components that compose the core functionality of COVID-X Sandbox.
- **Section 4** includes the final version of both local and centralized deployments of COVID-X Sandbox.
- **Section 5** provides technical instructions regarding the setup and configuration of the final version of COVID-X Sandbox and presents a guide for users on how they can access its services.
- **Section 6** concludes the document and describes all the inferences that came up regarding the design, development, and deployment of COVID-X Sandbox.

2 Sandbox CI / CD Stack – Updates and Final Release

As initially described in D2.1 [1] and further analysed in D2.2 [2], COVID-X Sandbox illustrates a set of services able to handle healthcare data sources and offer multiple services to third parties that cover data management, security, visualization, and federated learning. These services are realized as a collection of microservices that operate in parallel, forming a layered and modular architecture. This modular approach makes it essential for the COVID-X Sandbox to adopt the Continuous Integration / Continuous Deployment (CI/CD) practices and methods in order to validate that the various tools and components will be successfully integrated into the platform. Continuous Integration introduces an automated process that ensures code changes on a specific component/service of the Sandbox are sufficiently tested, and the new version is successfully integrated with the other components/services of the platform. On the other hand, Continuous Deployment is responsible for automatically deploying the latest version of the integrated platform to a pre-defined operational environment. Additionally, the CI/CD processes assist the third parties in regularly building, testing, and deploying their software solutions and applications into the COVID-X Sandbox.

The COVID-X Sandbox utilizes a set of open-source tools that form the Sandbox CI/CD Stack and are responsible for realizing the CI/CD methods and best practices. The CI/CD Stack has been successfully deployed by month 2 of the project and was presented in D2.2. This chapter presents an overview of the Sandbox CI/CD Stack, as well as the latest updates towards the Final Release.

2.1 Overview of the Sandbox CI/CD Stack

The Sandbox CI/CD Stack has been described in full detail in D2.2 [2]. For completeness purposes, and for the reader to have a better overview of the COVID-X Sandbox, this section provides a high-level overview of the CI/CD Stack, as it was developed and included in Release A.

As depicted in Figure 1, the Sandbox CI/CD stack comprises several open-source software tools, which collectively aim to create an automated build system capable of integrating changes performed by developers working on different components and services. The stack encapsulates the following software tools:

- **Gitlab**[1]: Gitlab is an open-source git-based Version Control System (VCS) responsible for accommodating the source code warehouses of the various Sandbox services. Under a private Gitlab group named "COVID-X-PRJ", several sub-groups and repositories that store the source code of the Sandbox services and the third-party solutions are created. Each partner has been granted access to the appropriate sub-groups to create repositories, organize users in teams and upload their source code.
- **Jenkins**[2]: Jenkins is an open-source tool acting as the CI/CD server, enforcing the automated building, testing, and deployment of software services. Some of the tasks that can be automated through Jenkins include software builds, unit testing, packaging, and pushing

[1] https://gitlab.com/
[2] https://www.jenkins.io/

container images to the Docker Registry. These tasks are implemented within specific Jenkins Pipelines created for each of the Sandbox components.

- **Docker**[3]: Docker is an open-source containerization platform enabling developers to package applications into containers. Docker is used as the standardized method for deploying the core Sandbox services, as well the third-party software solutions, into the Sandbox platform.
- **JFrog Docker Registry**[4]: JFrog is an open-source Docker registry acting as centralized storage and content delivery system that allows users to store, manage, and distribute named Docker images, available in different tagged versions. Developers and users of the Sandbox services can use this registry to push/pull their Docker images.
- **Portainer**[5]: Portainer is an open-source software tool offering a lightweight web User Interface (UI), allowing the Sandbox users and administrators to monitor and manage multiple Docker environments (docker hosts). More specifically, it can be used to manage all Docker resources (containers, images, volumes, networks, etc.), set up and manage multiple environments, deploy applications, monitor app performance, and triage problems.

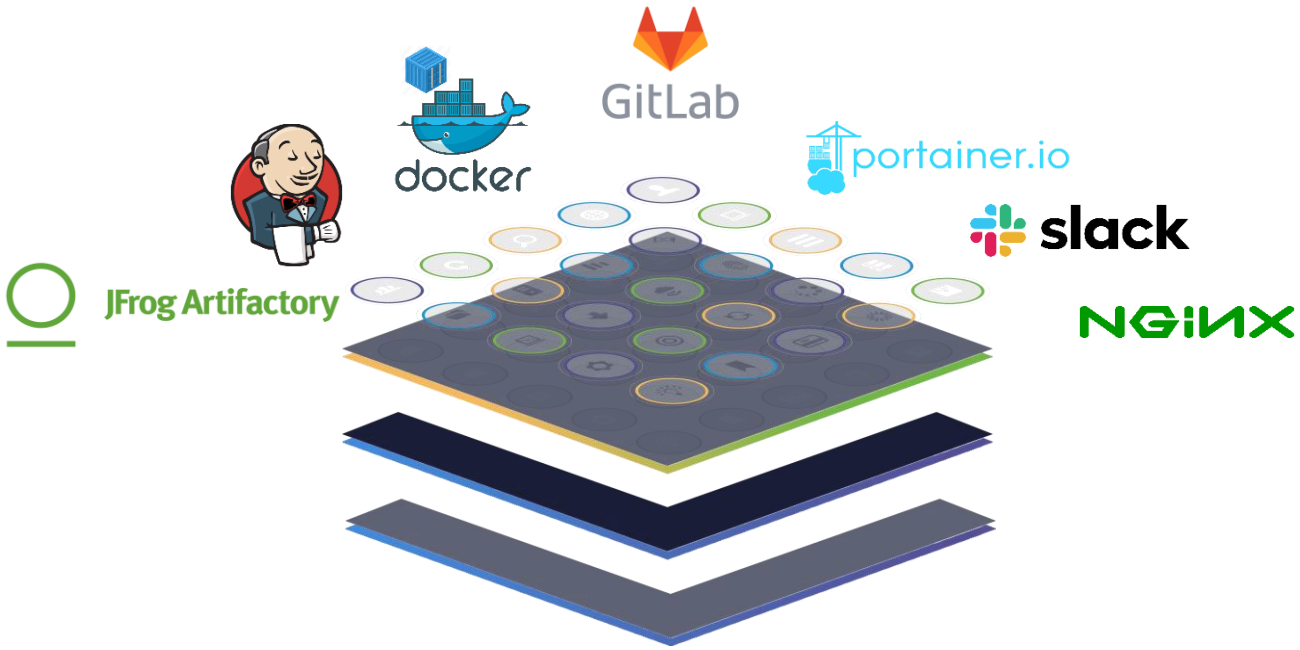

FIGURE 1: COVID-X SANDBOX CI/CD STACK

The software components comprising the COVID-X Platform are deployed on virtual hosts, which are cloud servers instantiated on Hetzner[6], an EU-based public cloud provider. These virtual servers are used to host the following Sandbox environments:

- **CI/CD Stack**: The software components comprising the CI/CD Stack, namely Jenkins, JFrog, Portainer, and Docker, are deployed on virtual servers.

[3] https://www.docker.com/
[4] https://jfrog.com/
[5] https://www.portainer.io/
[6] https://www.hetzner.com/

- **Development environment**: The Sandbox development environment consists of a set of virtual servers able to host one or more instances of the core Sandbox services for a short period. These temporary service instances are used for running all kinds of unit, functional, and integration tests defined by the Sandbox developers.
- **Deployment environment**: The Sandbox deployment environment consists of a set of virtual servers that can host an operational and stable version of the Sandbox services. This environment must support very high percentages of service uptime, since it is the one that third-party users and applications will be accessing for storing and retrieving healthcare data.
- **Third-party solutions environment**: The Sandbox third-party solutions environment consists of a set of virtual servers employed to host the software applications developed and integrated by the third-party teams joining the COVID-X project.

2.2 Updates and Final Release

This section presents the updates and advances of the Sandbox CI/CD Stack included in Release B.

2.2.1 Dedicated third-party solutions servers

In Release B of the COVID-X Sandbox, a dedicated virtual server is reserved and provided to each of the third-party teams that will integrate their software application into the Sandbox. Each team will use the server assigned to their organization to deploy, test, and incorporate their application with the Sandbox data management services. This approach further enhances the security and integrity of the applications developed and integrated by the third-party solutions into the COVID-X Sandbox, by creating isolated and fully protected environments. Additionally, the reservation and distribution of the Sandbox resources will be more efficient since they will be tailored to each application's specific needs and requirements.

Each server is configured to run a secured Docker Daemon, also accessible over the network to allow remote and automated deployment of dockerized applications. Furthermore, a Portainer service is installed on each server to ease the Docker resources management and monitoring for the team's developers and administrators. Effective firewall rules are configured on each virtual server to securely handle inbound and outbound network traffic, ensuring that only authorized users and applications are permitted to attain access. Finally, all the storage units (either ephemeral or persistent volumes) are encrypted to mitigate data exposure risks.

2.2.2 OpenVPN Server

As stated in the previous sections, the Sandbox environments are protected by firewall rules that permit inbound and outbound traffic only to and from specific static IP addresses. This minimizes the risk of an unauthorized user or application accessing the Sandbox servers. On the other hand, in some cases, it might not be possible for the Sandbox users to configure and maintain a static source IP address. In order to cover the latter scenario, the COVID-X Sandbox Release B supports an OpenVPN server, allowing users to connect via a secured Virtual Private Network (VPN) tunnel to the Sandbox premises.

When establishing a VPN connection between two sides, a private overlay network is created through the public Internet, enabling these two sides to communicate directly. The private overlay network is called VPN tunnel and is responsible for forwarding encrypted network traffic. OpenVPN is both an open-source VPN protocol and a VPN software that enables users and applications to create secured VPN connections. The Sandbox CI/CD includes an OpenVPN server, where users and applications can be connected using their specific user/password credentials along with a client-side certificate, getting access to the Sandbox private environments.

The Sandbox utilizes PFSense[7], an open-source software firewall, to implement the OpenVPN server. PFSense supports various VPN solutions, including OpenVPN. Additionally, being a software firewall, PFSense enhances the Sandbox security by design. PFSense runs on a dedicated virtual server of the Hetzner cloud, through which clients can access the rest of the Sandbox services. Through the PFSense web UI, the OpenVPN is configured, as depicted in the Figure below.

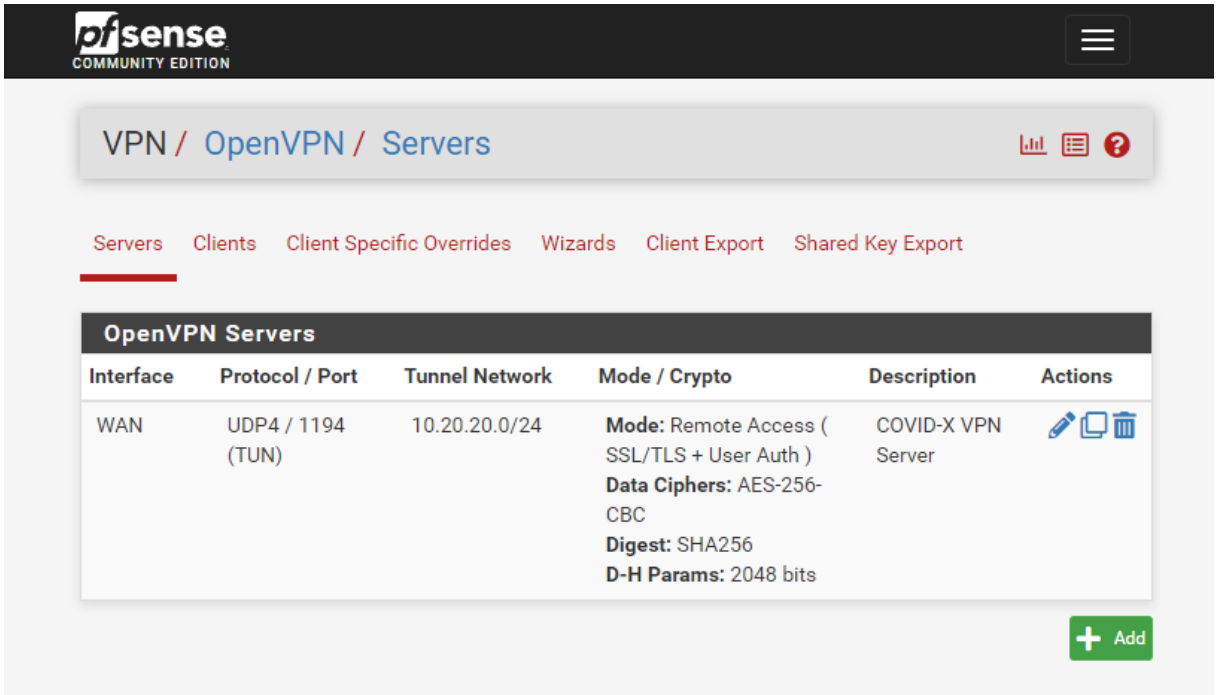

FIGURE 2: COVID-X SANDBOX PFSENSE

[7] https://www.pfsense.org/

3 Sandbox Integrated Services – Release B

This section describes all the updates in the implementation of COVID-X Sandbox, compared to the first release presented in D2.2 [2]. The first subsection provides an overview of the final version of COVID-X Sandbox, while all the other subsections describe any updates or additions in terms of the technologies and applications that have been initially introduced to develop COVID-X Sandbox services.

3.1 Update of the Sandbox Architecture – Release B

3.1.1 COVID-X Sandbox Architecture

COVID-X Sandbox is built as a collection of microservices, following the Service-Oriented Design approach, as initially described in D2.1 [1]. Like the first release, COVID-X Sandbox contains a set of (i) Data Management, Harmonization and Visualization services, (ii) Security services and (iii) Sandbox interfaces. In addition to the first release, federated learning (FL) services are introduced and implemented, as part of the final release of COVID-X Sandbox. FL services are further presented and analysed in section 3.4 of the current document. All these services follow the CI/CD workflow enabling technical teams to contribute independently in the development phase of these services.

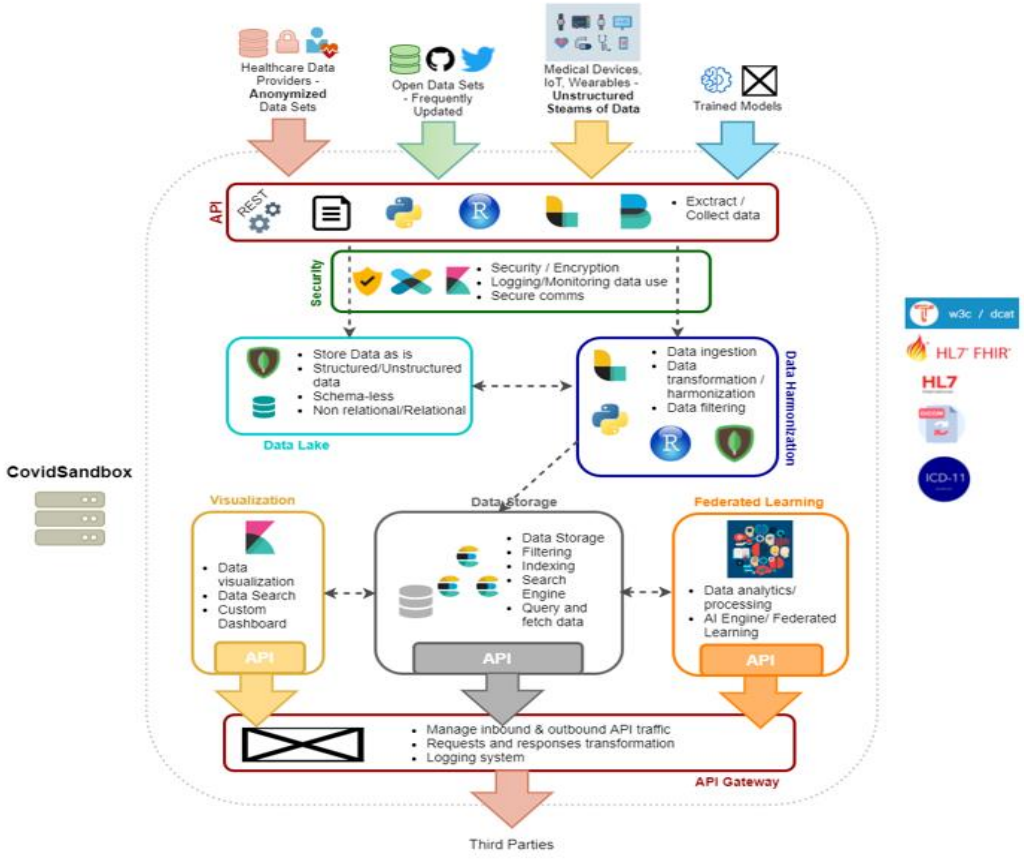

FIGURE 3: COVID-X ARCHITECTURE - RELEASE B

These services are integrated and combined to form the Sandbox architecture for the final release on month 12 of the project. Communication between these services is achieved through REST endpoints exposed internally, based on the technologies used for the implementation of each service independently. Similar to D2.2 [2], specific HTTP methods (GET, POST, PUT, DELETE) allow client services to access COVID-X Sandbox services' resources, and data exchange is achieved with information represented in a structured text format (JSON). The figure above presents the overview of the final system architecture of COVID-X Sandbox.

3.1.2 COVID-X Semantic Data Model

In COVID-X, we aim to collect data from a multitude of third-party organizations, store them in a unified manner and make them accessible to the corresponding third parties for further visualization and management. In the current section, we demonstrate the work done towards the creation of a unified data model that captures the main characteristics common to the datasets included in the COVID-X Sandbox.

Overview

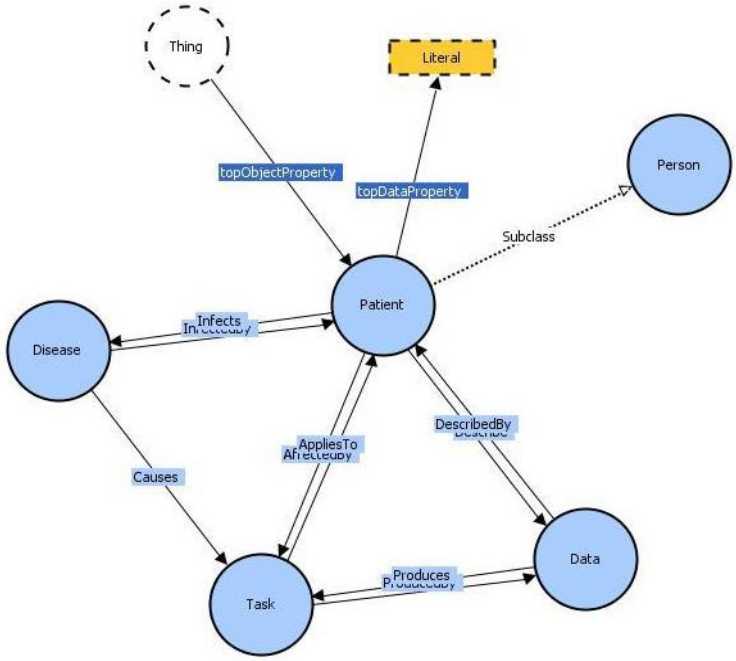

FIGURE 4: THE TOP LEVEL HIERARCHY OF THE UNIFIED DATA MODEL IN PROTEGÉ

First, we created the upper-level classes of the data model hierarchy (see Figure 4). These are the following: 1) Person, 2) Disease, 3) Data and 4) Task. The patient is infected by a Disease, which in our case is SARSCov2 (ICD-11 code: XN109SARSCov2). The patient is described by a set of data which is further categorized as presented in Figure 5 below, and this data produces a set of Tasks to be applied to the patient.

Patient

Through studying all datasets included in the COVID-X Sandbox we identified a set of variables common to most of them, which may form a complete patient profile. The selected variables to form the Patient Profile in the Unified Data Model can be seen in Figure 2.

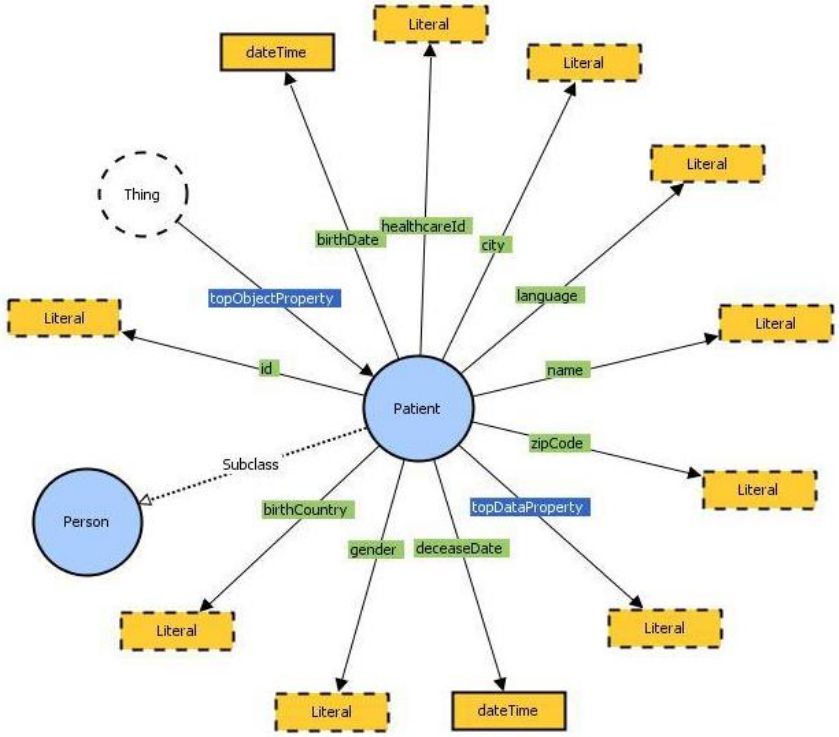

FIGURE 5: THE SUBCLASSES OF THE PATIENT CLASS IN PROTEGÉ.

Disease

In Figure 6, the relationships between the Disease Class and its ICD-11 analogue are presented. ICD-11 constitutes a revision to the ICD-10 coding; it provides a more sophisticated structure to classify diseases, disorders, injuries, and causes of death, with around 55 000 codes that offer a fine level of detail in coding these illnesses. Apart from the ICD-11 coding of the reference disease, we provide a mapping between the various symptoms included in the COVID-X Sandbox datasets and presented in Figure 5 and their ICD-11 codes. An example of this mapping is presented in Table 1 below.

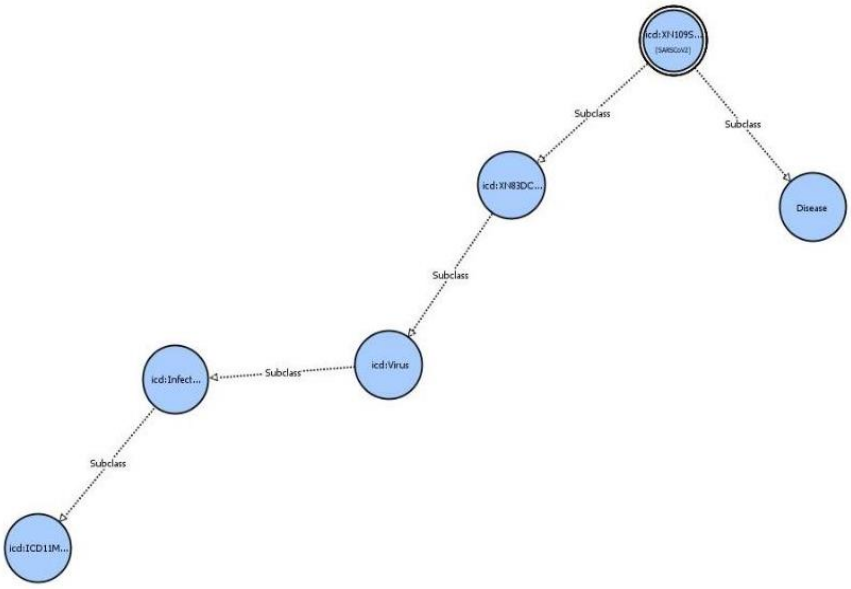

FIGURE 6: THE REPRESENTATION OF THE DISEASE USING ITS ICD-11 CODE IN PROTEGÉ.

Symptom name in plain text	ICD-11 code of symptom
abdominalPain	MD81
breathShortness	MD11
chestPain	MD30
chills	MG21
cough	MD12
emesisOrNausea	MD90
jointPain	ME82
legPain	ME86.B
myalgia	FB56.2
nasalCongestion	MD34
soreThroat	MD36

TABLE 1: SYMPTOMS MAPPING BETWEEN UNIFIED SEMANTIC DATA MODEL AND ICD-11 CODES

The mapping of the variables composing the COVID-X Unified Data Model to their ICD-11 codes is a crucial step towards achieving interoperability of the datasets included in the COVID-X Sandbox with other relevant datasets. Thereby, COVID-X makes an important step towards promoting the FAIR data principles by means of transforming the datasets included in the COVID-X Sandbox to comply with the FAIR data principles.

Task

The Task class is further divided in the following Subclasses: 1) Clinical (Diagnosis, Mechanical Ventilation & Therapy), 2) Administrative (Admission, Hospitalization & Transfer) and 3) Interview (Questionnaire & Chatbot). The 1st subclass refers to Clinical Tasks that involve the Patient in an active (Therapy the Patient needs to have, the Patient is actively involved in e.g., taking their own medicine, etc.) or passive manner (Diagnosis made based on the Patient's examinations). The 2nd subclass describes a 3-stage Administrative Task structure that the Patient is involved in. This has to do with a Patient's Admission to the hospital, their Hospitalization period and possible intra- or inter-hospital Transfer. The third Subclass refers to the Interviews the Patient may need to complete throughout the whole course of the Disease. These Interviews can either have the form of Questionnaires containing specific Questions or they could be executed through a Chatbot. The Subclasses and Individuals comprising the Task Class are presented in Figure 7.

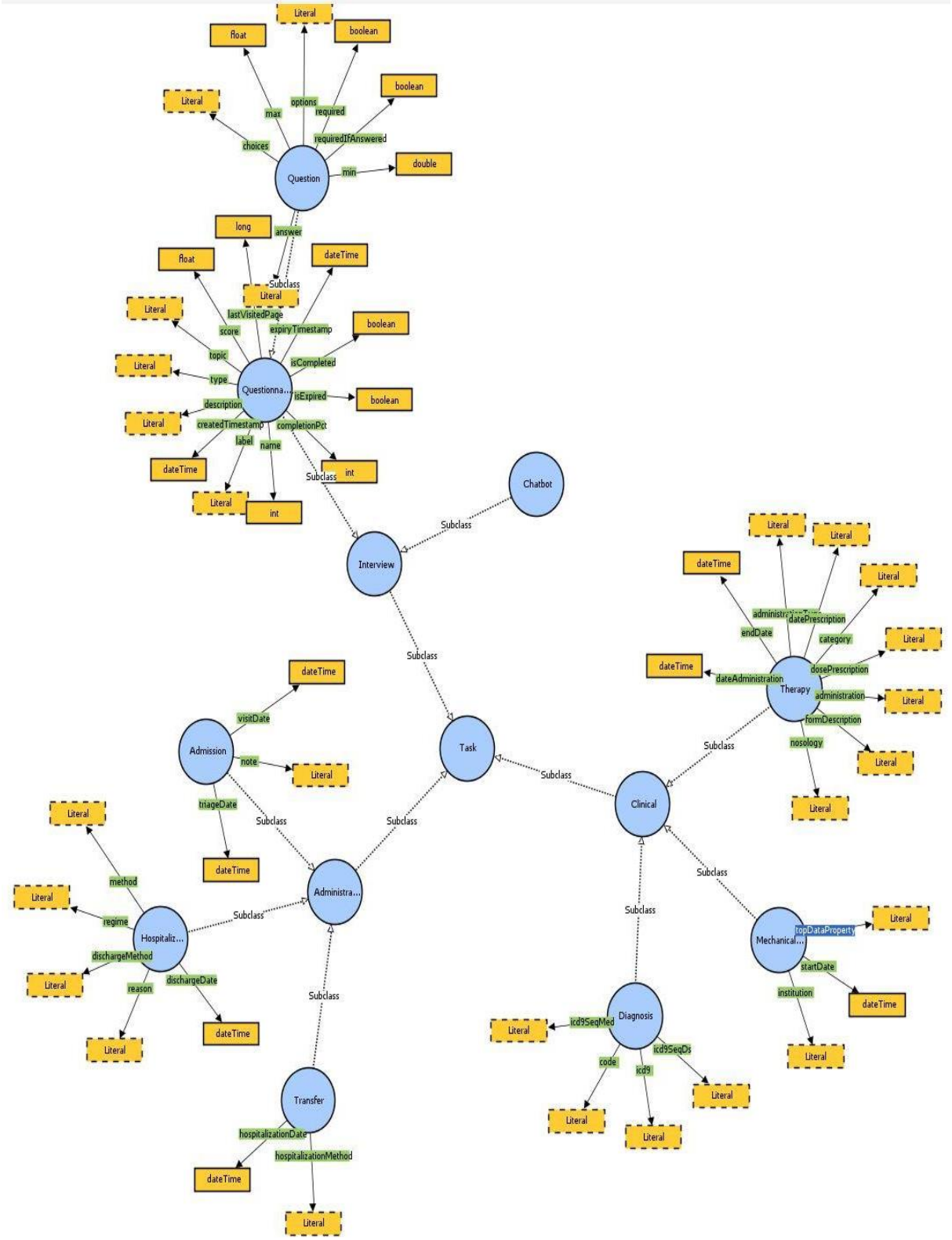

FIGURE 7: THE SUBCLASSES AND INDIVIDUALS OF THE TASK CLASS IN PROTEGÉ

Data

The Data Class hierarchy is presented in Figure 8. As can be seen, it is further divided in 1) Medical and 2) Statistical Data. The 1st Subclass is divided in 1) Device – referring to the Characteristics or Specs of

the Device used in monitoring a Patient, 2) History – referring to the Medical History of the Patient (*neurologicalDiseases*, *renalDiseases*, *dataProvenance*, *allergies*, *bloodType*, *asthma* etc.), 3) Lab – referring to the laboratory tests conducted for a Patient and containing ID, result, units, valMin and valMax, allowing data transformation to achieve homogenization of the results, 4) Monitoring – referring to all parameters relevant to the Monitoring of a Patient, that are Activity Monitoring (*stepCounts*, *distanceWalkRun*, etc.), Rehabilitation, Signs of the Disease (*SpO_2, respirationEvaluation, restingHeartRate, BMI* etc.) and Symptoms (*diarrhea*, *dizziness, jointPain, myalgia, nasalCongestion, rhinorrhea, smellLoss, soreThroat, tasteLoss* etc.) and 5) Note – referring to Notes kept in free text regarding a Patient's situation. The 2nd Subclass, Statistical Data, refers to the Open-Source Datasets ingested in the COVID-X Sandbox, containing either Reports (i.e., Demographic Data) or Medical Images (CT scans, X-rays, etc.).

Moreover, after studying all variables contained in all datasets, we realized some variables referring to the same metric but named differently, e.g., *restHeartRate / restingHeartrate, temp / temperature / body_temperature, respEval / respiration_evaluation, SpO_2 / oxygenSaturation*, etc.). In order to further enhance data interoperability, we selected for each variable the most representative name (i.e., *respiratoryEvaluation* instead of *respEval*) and we adopted the Camel Case Naming Convention for the variables' names, i.e., we keep the first letter lowercase, while the first letter of each internal word capitalized (***s**mell**L**oss*, ***r**esting**H**eart**R**ate*, etc.).

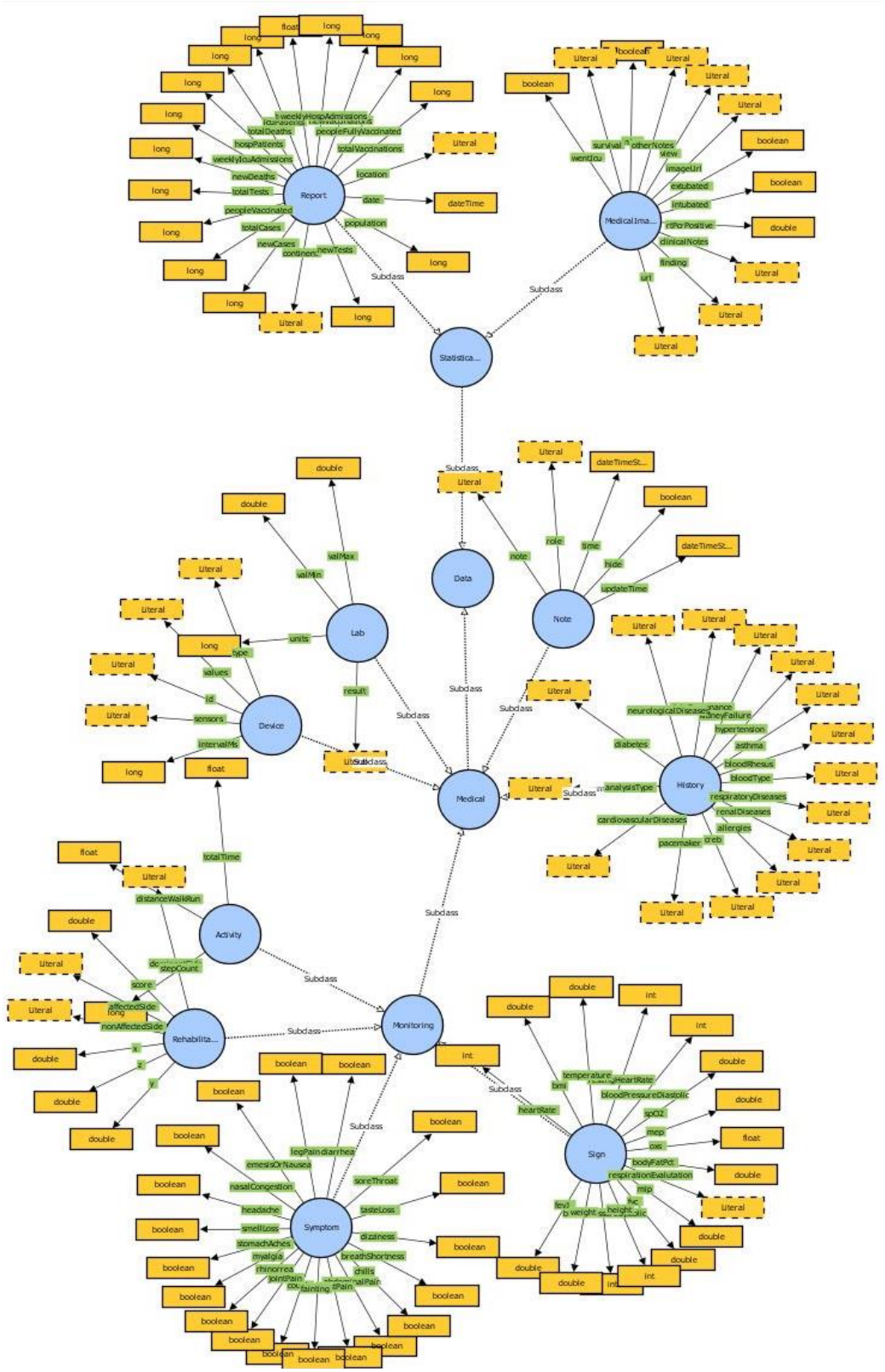

FIGURE 8: THE SUBCLASSES AND INDIVIDUALS OF THE DATA CLASS IN PROTEGÉ

3.2 Data Integration, Management, Harmonization and Visualization Services

3.2.1 Logstash

As presented in D2.2 [2], Logstash is the main technology used for creating pipelines responsible for collecting and integrating data from healthcare data sources before storing in the Data Storage layer in COVID-X Sandbox. Following the deployment of the first version of COVID-X Sandbox and as part of the process of OC1 solutions integration, several data pipelines have been implemented to ingest all the information from data providers to support third parties' needs.

Logstash has three areas of function:

- Process input of the log file records from the remote source with Logstash input plugin in the pipeline configuration file
- Transform and enrich the log records with Logstash filter plugin in the pipeline configuration file
- Send the log records to the destination using output plugin of the Logstash pipeline configuration file

With Logstash input plugin, it is easy to define various methods to receive data records from many sources, e.g. reading from a flat file or from a TCP port or directly from a remote or local syslog. In the most common case, when Logstash reads from a flat file, it is necessary to define the path of the local filesystem where Logstash is set up. Logstash polls constantly this path for any new files and then it starts passing its content line by line into the next step of processing. Another input plugin commonly used is the http plugin, where Logstash receives events over HTTP or HTTPs upon user's request. This plugin is very useful when the input message is a multiline JSON including either a single JSON document or an array of JSON documents. Another useful input plugin used for polling HTTP APIs and directly fetching records is the http_poller plugin. This plugin periodically calls a specific URL address and gets its content as a response for further processing.

Next, the processed event can be mutated to alter it, remove dispensable data, or add additional information. Logstash provides plugins for many different data operations. Filters are often applied conditionally and can perform complex operations. For example, with grok filter it is possible to extract data from a string field containing text with a known pattern and with ruby filter it is possible to execute custom ruby code for event processing.

```
input {
    http {
        port => 8083
        user => "${ELASTIC_USERNAME}"
        password => "${ELASTIC_PASSWORD}"
        ssl => false
        codec => json
    }
}

filter {
    json {
        source => "message"
        target => "data"
    }
    mutate {
        convert => {
            "[payload][interval_ms]" => "float"
            "[payload][battery]" => "integer"
        }
        remove_field => ["message", "@version", "headers", "host"]
        add_field => { "[@metadata][index]" => "aviate.sensors" }
    }

    ruby {
        code => "
            event.set('[payload][values]', event.get('[payload][values]').map { |inner|
                    inner.each_index { |i|
                        inner[i] = inner[i].to_f
                    }
                    inner
                }
            )
        "
    }

}

output {
    elasticsearch {
            hosts =>  "${ELASTICSEARCH_HOST_PORT}"
            user => "${ELASTIC_USERNAME}"
            password => "${ELASTIC_PASSWORD}"
            ssl => true
            ssl_certificate_verification => false
            keystore => "/usr/share/logstash/config/certs/elastic-cert.p12"
            keystore_password => "${ES_KEYSTORE_PASSWD}"
            truststore => "/usr/share/logstash/config/certs/elastic-ca.p12"
            truststore_password => "${ES_TRUSTSTORE_PASSWD}"
            http_compression => false
            index => "%{[@metadata][index]}"
    }

    #stdout{}
}
```

FIGURE 9: LOGSTASH PIPELINE CONFIGURATION EXAMPLE

Finally, when outputting data, Logstash supports a huge range of destination types. Usually, all events are sent to Elasticsearch, but Logstash can be used also independently to save data to a CSV file, an SQL database, data analytics algorithm or just showing it to the console for debugging purposes. In terms of COVID-X sandbox, the main plugin used for output processed events is the Elasticsearch plugin. During the configuration of this plugin, we set up all the necessary information regarding the Elasticsearch host, the users with the proper rights to create and insert data into Elasticsearch, and finally the index name where data will be flushed.

3.2.2 Elasticsearch

Elasticsearch, as initially introduced in D2.2 [2], is an open source, distributable and highly scalable search engine designed to have optimal performance. It has a comprehensive REST-based API which allows to monitor and control all aspects of the cluster configuration. It provides endpoints to perform create, update, retrieve and delete operations on stored data over HTTP API calls. To some extent, Elasticsearch can be used like any other NoSQL data store as it contains a JSON-based query language to retrieve data.

Update process in a single document on Elasticsearch index is expensive in terms of performance. Therefore, if a data retention mechanism is required, it is recommended to split data into multiple indices based on timestamp. Delete operation is performed on the whole index. Depending on the number of data, indices can be created on a monthly, weekly, daily, or even hourly basis.

In terms of COVID-X sandbox cases, open datasets represent a typical case where data retention is necessary because the schema does not change, and we need to keep all the updated version of data across the time. So, Logstash configuration files implementing ingestion pipelines for open datasets, flush the processed events into an Elasticsearch index whose name contains current day's timestamp as its suffix.

Elasticsearch is a schema-less database, sufficient to send JSON objects as its input and it creates a document with the appropriate mapping for each JSON field automatically, with no performance overhead. Although it is also possible to define mapping implicitly, it must be done before adding the first value of that type, because it is not allowed to alter existing field mappings.

Elasticsearch provides a dedicated API for mapping configuration. The default type mapping has reasonable default values, but when you want to change default mapping or customized indexing, you need to own a mapping definition. By default, dynamic mapping is enabled. To prevent unnecessary indexing, it is possible to disable dynamic mapping for selected parts of document – in this case inner fields without static mapping will be skipped. In some cases, Elasticsearch cannot deduct automatically correct mapping; so, if we know the purpose of a specific field, we can prevent overhead and define in advance how it will be indexed.

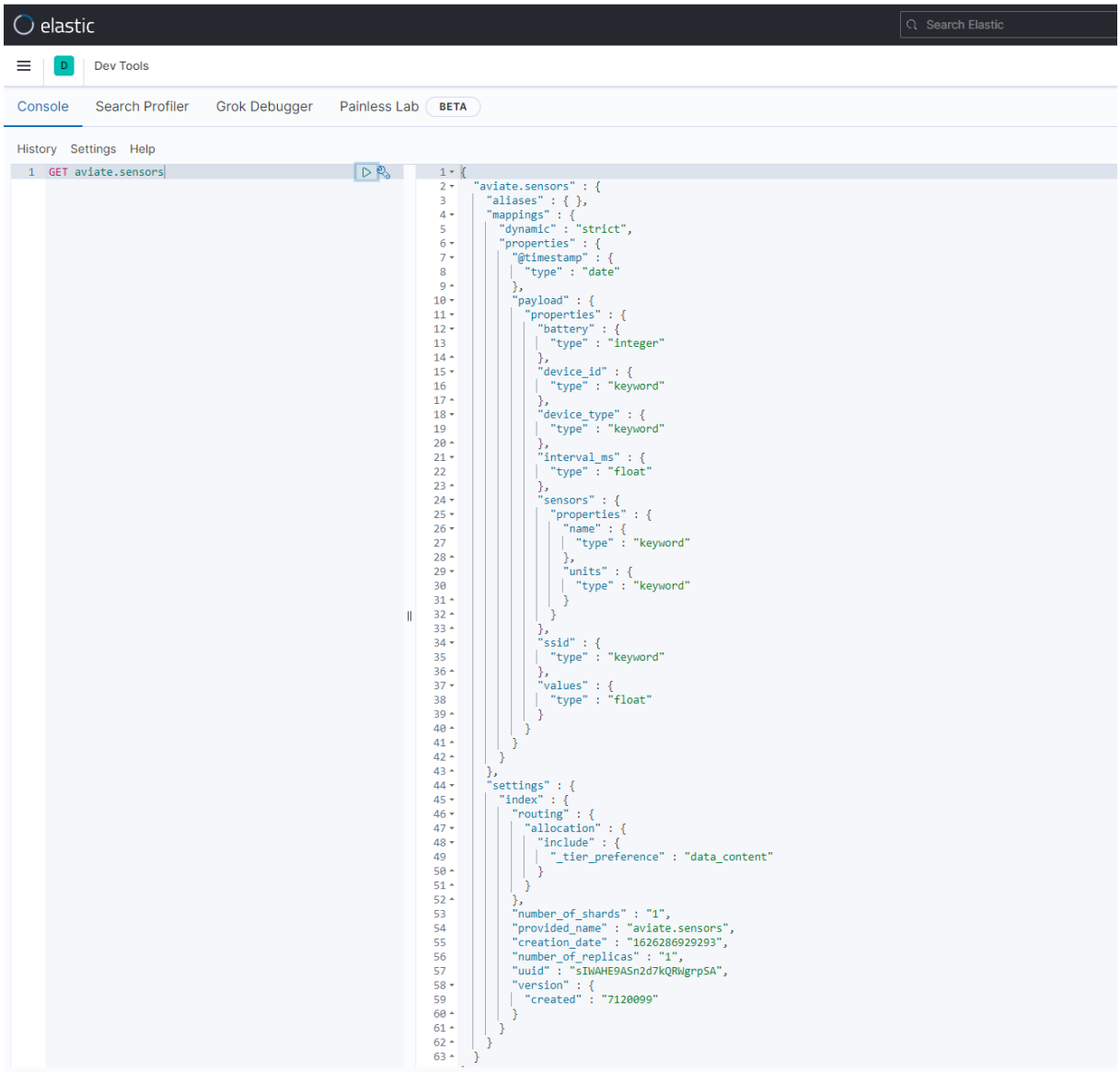

FIGURE 10 : ELASTICSEARCH INDEX MAPPING EXAMPLE

The mapping can be created together with the index, but this might be a bit problematic when Logstash creates new indices based on selected patterns, e.g. new index on daily basis.

3.2.3 Kibana

Kibana, as part of the ELK stack, is a proprietary data visualization tool used for interpreting and representing indexed Elasticsearch data in a graphical way. The main view of Kibana is divided into four main components – *Discover*, *Visualize*, *Dashboards* and *Management*.

The *Management* section is where all Kibana internals can be configured. It is where index patterns must be set. If index patterns contain time-based events, it is also needed to specify the timestamp field based on which Kibana will sort and filter data. Kibana, based on a set of indices which fulfil selected index patterns, shows the list of index fields together with their type and properties. In addition, it is possible to modify field formatters to alter how the field value is shown in Kibana GUI.

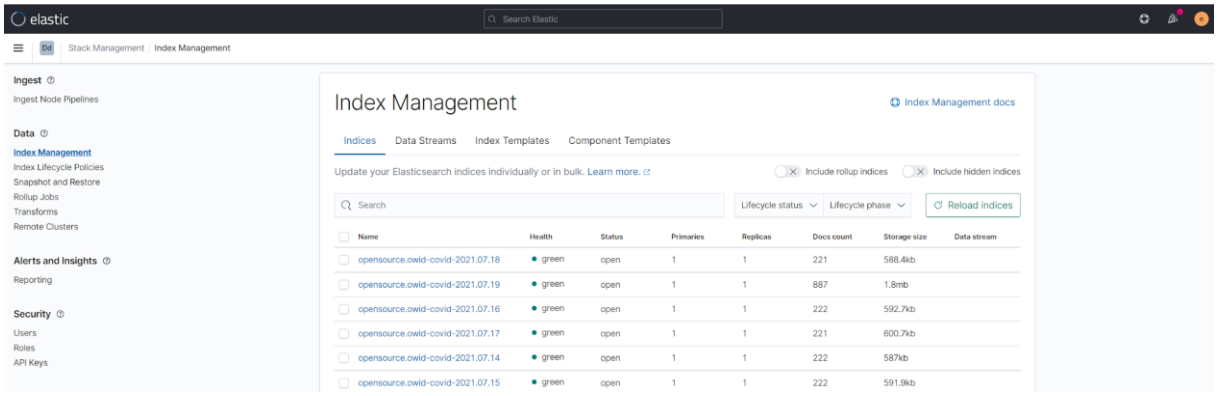

FIGURE 11: KIBANA - INDEX MANAGEMENT SECTION

Discover allows us to interactively browse and analyse data entries. Elasticsearch documents are categorized based on Index patterns defined in the *Management* section. They can be easily searched and filtered by time or document properties using Apache Lucene query syntax. One can also see how many documents match the search query or get field value statistics.

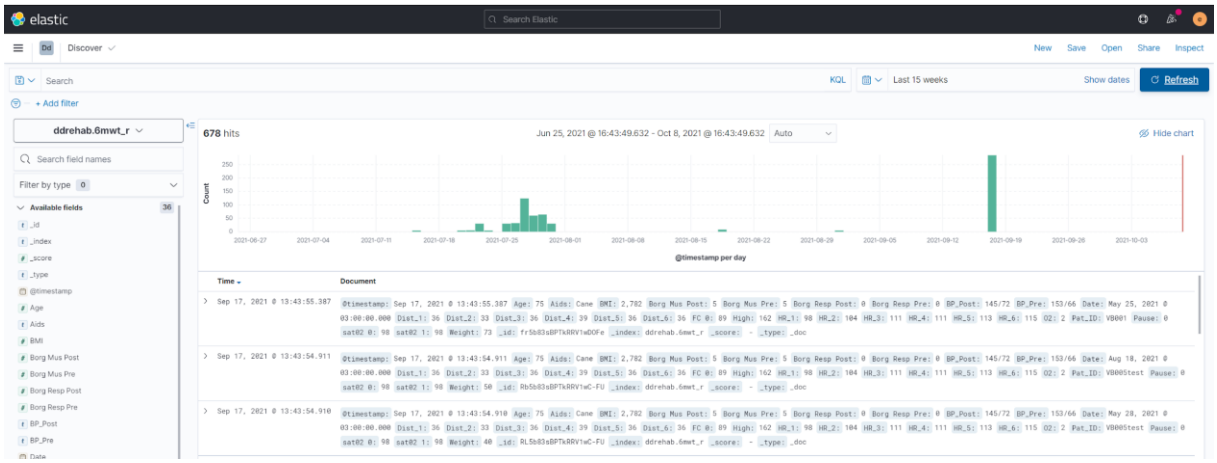

FIGURE 12: KIBANA - DISCOVER SECTION

Due to plugin-based architecture, Kibana can be easily extended to suit particular needs. Visualize section provides the possibility to visualize data with one provided visualization plugins. Information can be shown as tables, charts, histograms, and many others. Large volumes can be easily shown as pie, bar, line or scatter charts. Unfortunately, functionality to export raw data from Elasticsearch is missing.

Dashboard allows to combine multiple saved *Discover* and *Visualize* results into one view. It is possible to arrange and resize the dashboard elements as needed. Having some experience, it is possible to create very sophisticated and colourful dashboards, directly from Kibana GUI. Dashboards can be easily saved, shared or embedded into another web page.

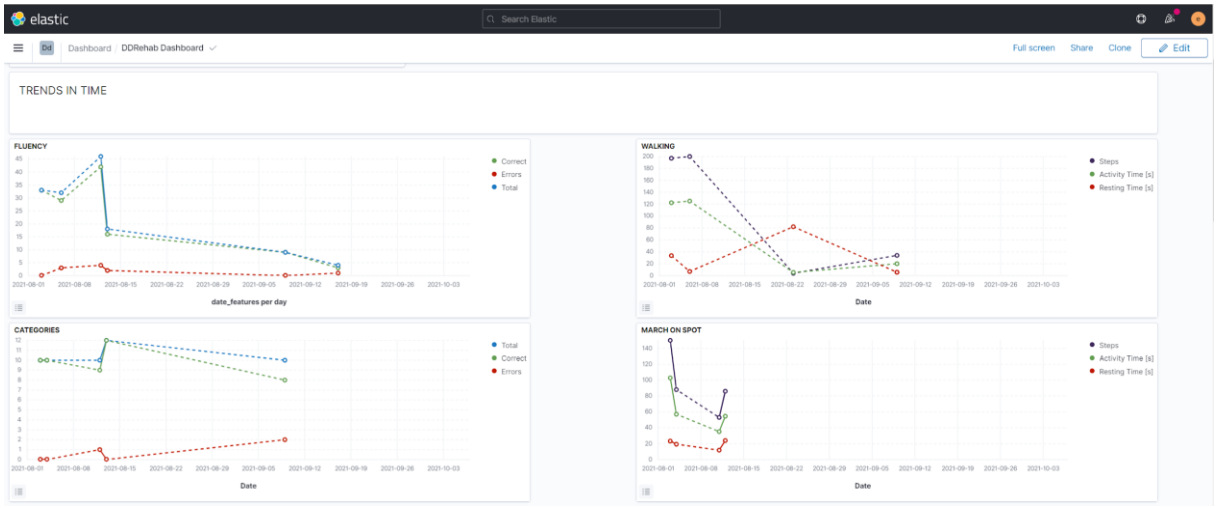

FIGURE 13: KIBANA - DASHBOARD EXAMPLE

The main drawback of Kibana is not having out of the box user management. The simplest solution to restrict access to selected users is to use Nginx proxy forwarder. If more elaborate access control is required, it is possible to purchase commercial support from Elastic to get a license from X-Pack – this will not only allow configuring sophisticated security options, but also enable reporting and alerting features.

3.2.4 MongoDB

MongoDB[8] is an open-source NoSQL Database Management System (DBMS) that uses JSON-like documents instead of relational tables and rows to process and store various forms of data. It allows dynamic schema design and introduces a structured query language to access stored data. MongoDB supports replica sets for high availability, allowing data stored in a primary instance to be duplicated to secondary stores for reading purposes.

The COVID-X Sandbox capitalizes on MongoDB to create a key-value repository for storing data and information related to the Sandbox deployment. More specifically, it keeps information related to the nodes that form the ELK cluster, such as the ID, IP address, physical host, and available resources of each node. The Sandbox reads and updates this information every time a node joins or leaves the ELK cluster. This process empowers the Sandbox to efficiently manage the ELK's multiple nodes ; allowing multiple nodes to be dynamically added to the ELK cluster without interrupting the operational functionalities or affecting the uptime of the Sandbox. Furthermore, it enables the ELK to cluster nodes to be distributed on multiple physical hosts of the system, offering high availability, fault tolerance, flexibility, and increased performance capabilities.

MongoDB is deployed in a Docker container running as part of the overall Sandbox microservices Stack. The MongoDB container is attached to the same Docker network used by the other Sandbox services, enabling communication channel amongst them. Whenever a node joins or leaves the ELK cluster, the

[8] https://www.mongodb.com/

Sandbox reads and updates the respective information in the MongoDB documents through a MongoDB Python client function[9].

3.3 Security Services

The COVID-X Sandbox was designed and implemented in a privacy-by-design way, by capitalizing on the security features offered by Elastic Stack (ELK Stack). The security services of the COVID-X Sandbox ensure three main security principles: Confidentiality, Integrity and Availability (CIA) of the data stored within the Sandbox. These are ensured through the implementation of **authentication**, **authorization** and **encryption** mechanisms, as well as **clustering** capabilities supported by ELK. In Table 2, we see the correspondence between security features and CIA principles.

Feature Description CIA	Description	CIA
Authentication	Verification the user or entity is who they claim to be	Confidentiality
Authorization	Once authenticated, data the user/entity can access and operations they can perform on this data	Confidentiality, Integrity
Encryption	Transform data-in transit making it inaccessible to unauthorized users.	Confidentiality, Integrity
Clustering	Group of independent servers connected via network share computing tasks	Availability

TABLE 2: SECURITY SERVICES - FEATURES DESCRIPTION

The above security measures intend to satisfy all applying national and European regulations with regards to usage and storage of personal data, like the GDPR.

With regards to user authentication: ELK Stack authenticates users by identifying who is making a request and verifying their identity. The Elastic Stack security features authenticate users by using realms and one or more token-based authentication services. A *realm* is used to resolve and authenticate users based on authentication tokens. ELK has multiple industry standard realms built-in, like Active Directory and LDAP, while it also offers the Elasticsearch native realm, which is the default. We have used the native real for the two COVID-X Sandbox releases. The native realm is an internal one where users are stored in a dedicated Elasticsearch index. This realm supports an authentication token in the form of username and password.

Moreover, for the second COVID-X Sandbox release, an OpenVPN server was set up to allow external partners to access services and applications hosted on the Hetzner public cloud using the PFSense software firewall. This enables further protection of the hosted services through the configuration of iptables firewall rules, permitting access only to specific source IP addresses. With the use of the OpenVPN server, external partners can reach the ELK services through a private and encrypted VPN tunnel.

[9] https://www.mongodb.com/languages/python

With regards to user authorization: Within the ELK Stack, it is achieved through a role-based access control (RBAC) mechanism that determines whether the user behind an incoming request is allowed to execute it. The internal native realm enables to manage and authorize users. The Kibana interface, along with the elastic API, provide ways to add and remove users, manage their passwords, assign them specific roles and bind these roles to specific permissions.

The steps followed to set up the RBAC system for the COVID-X Sandbox were the following[10]:

1. Index pattern creation: First, we created index patterns for each OC! third party, whose data were to be ingested in the Sandbox. Additionally, we created two more index patterns: one to act as a data catalogue and another to accommodate the open-source data from the selected public sources (see Figure 14).

```
payload = {"refresh_fields": 'true', "index_pattern": {"id": "indexPatternX", "title": index_pattern}}
payloadJson = json.dumps(payload)
requests.post('https://dmht-kibana:5601/api/index_patterns/index_pattern', auth=(superuser, password), data=payloadJson, headers={'kbn-xsrf': 'true', 'Content-Type': 'application/json'}, verify=False)
```

2. Kibana space creation: Spaces enable to meaningfully categorize and structure the offered dashboards. Within the COVID-X Sandbox, we created one space per third-party to provide total isolation among data of various third parties. Once inside a space, one can only see that space's dashboards. Furthermore, we control which features are visible in each space. Of course, Kibana spaces are not meant to provide security; they are just meant to provide more intuitive and well-structured visualizations. In order to ensure security, we created roles granting users the desired Kibana privileges granting users access to a subset of spaces or features (see Figures 15-16).

```
payload = {"id": "spaceIdX", "name": "SpaceIdX", "description": "This is the SpaceIdX Space", "color": "#aabbcc", "initials": "X", "disabledFeatures": ["savedObjectsManagement", "savedObjectsTagging", "enterpriseSearch", "logs", "infrastructure", "apm", "uptime", "siem", "dev_tools", "advancedSettings", "fleet", "actions", "stackAlerts", "monitoring"], "imageUrl": ""}
payloadJson = json.dumps(payload)
requests.post('https://dmht-kibana:5601/api/spaces/space', auth=(superuser, password), data=payloadJson, headers={'kbn-xsrf': 'true', 'Content-Type': 'application/json'}, verify=False)
```

3. Role creation: Next, we created roles for each group of users and assigned them different index and Kibana privileges. More specifically, we created two kinds of roles for users of each third party: one administrative role, with *read* and *write* privileges to the respective third-party

[10] The whole source code for the RBAC provision can be found in the COVID-X Gitlab.

indices and one *read-only* role for the respective third-party indices. *It is important to notice that no user from a certain third-party organization has privileges to read and/or write data belonging to another third-party organizations, except for the System Owner.* Moreover, apart from the privileges on the respective organization's indices, all roles have *read-only* privileges in the *data_catalogue* and the *opensource* indices, as well as the *view_index_metadata* privilege to all indices. In addition to custom roles, ELK has some built-in roles (apm_system, apm_user, beats_admin, beats_system) reserved to ELK, but not used for the 1st and 2nd releases of the Covid-X Sandbox (see Figures 17-19).

```
payload = {"elasticsearch": {"cluster": [],   "indices": [{
        "names": users["spaceIdX"]["kib_index_pattern"],
        "privileges": ["read", "write", "view_index_metadata"]
    }, {
        "names": common_indices,
        "privileges": ["read", "view_index_metadata"]
    }
    ]
    },
        "kibana": [
          {
            "base": [],
            "feature": {"discover": ["all"],
                    "dashboard": ["all"],
                    "canvas": ["all"],
                    "maps": ["all"],
                    "ml": ["all"],
                    "visualize": ["all"],
                    "indexPatterns": ["all"]},
                    "spaces": ["spaceIdX"]
          }
    ]
}
payloadJson = json.dumps(payload)
requests.put(https://dmht-kibana:5601/api/security/role/RWRoleName', auth=(superuser, password), data=payloadJson, headers={'kbn-xsrf': 'true', 'Content-Type': 'application/json'}, verify=False)
```

4. Security privileges: All previously created roles assigned with some privileges.
    a. Index privileges:
        i. *read*: Read-only access to actions (count, explain, get, mget, get indexed scripts, more like this, multi percolate/search/termvector, percolate, scroll, clear_scroll, search, suggest, tv).

ii. *write*: Privilege to perform all write operations on documents, including permission to index, update, and delete documents, as well as performing bulk operations, and allowing dynamic mapping updates as a result.

iii. *view_index_metadata*: Read-only access to index and data stream metadata (aliases, exists, field capabilities, field mappings, get index, get data stream, ilm explain, mappings, search shards, settings, validate query). This privilege is primarily available for Kibana users[11].

b. Kibana privileges: Grant users access to features within Kibana. Roles have privileges to determine whether users have *write* or *read* access within a specific Kibana space.

i. *all*: Grants full read-write access.

ii. *read*: Grants read-only access.

5. User creation: Then, we created users for each third-party. This was done in coordination with the third parties, who provided users' names, emails, and desired roles for each user. We used this input to create users and assign their appropriate roles, ensuring them the necessary privileges s to do exactly and only what they are entitled to.

```
payload = {"password": info["password"],
                "roles": space.lower() + info["role"],
                "full_name": info["Fullname"],
                "email": info["email"],
                "metadata": {}
            }
payloadJson = json.dumps(payload)
requests.post('https://dmht-elastic:9200/_security/user/userX', auth=(superuser, password), data=payloadJson, headers={'kbn-xsrf': 'true', 'Content-Type': 'application/json'}, verify=False)
```

[11] https://www.elastic.co/guide/en/elasticsearch/reference/current/security-privileges.html

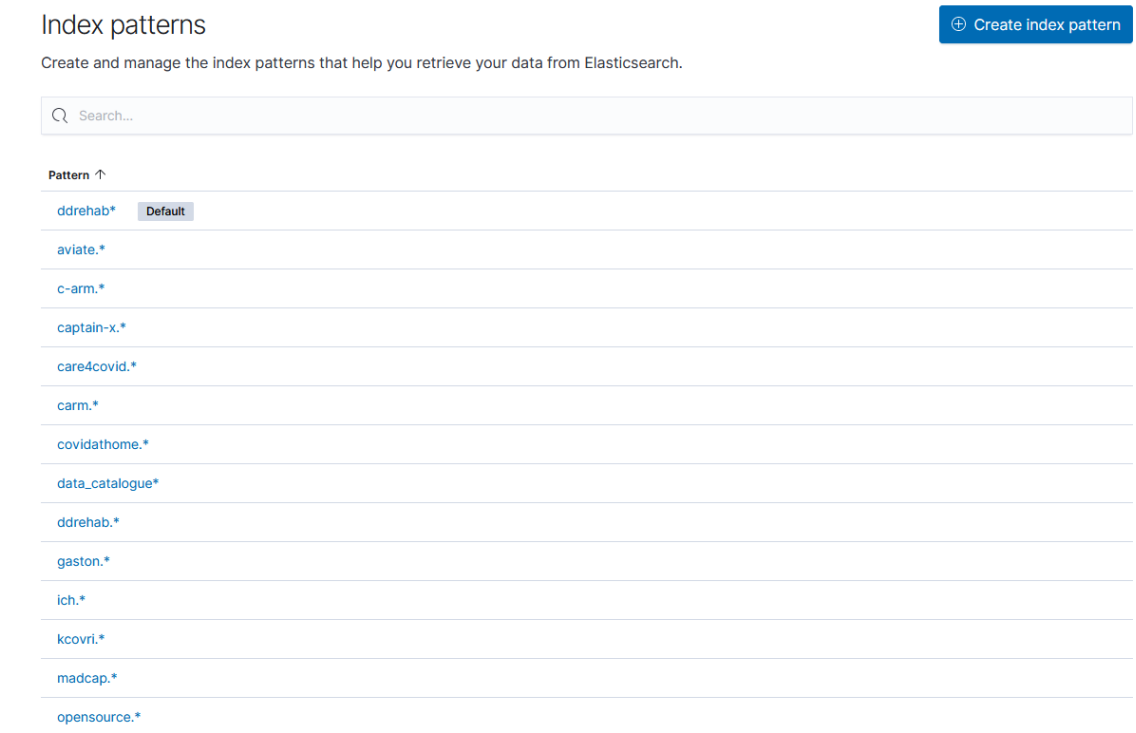

FIGURE 14: ELASTICSEARCH INDEX PATTERNS (COVID-X SANDBOX - OPEN CALL 1)

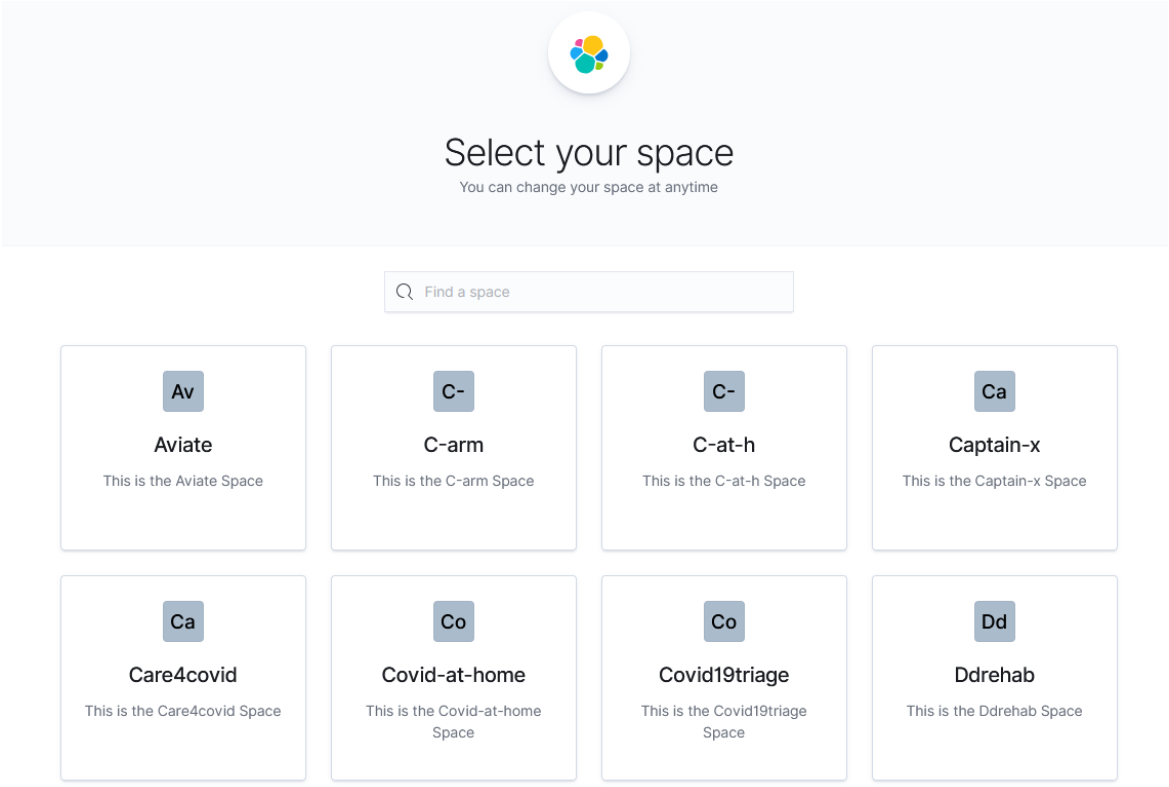

FIGURE 15: KIBANA SPACE SELECTOR VIEW

When logging in to Kibana, one is asked to choose a space.

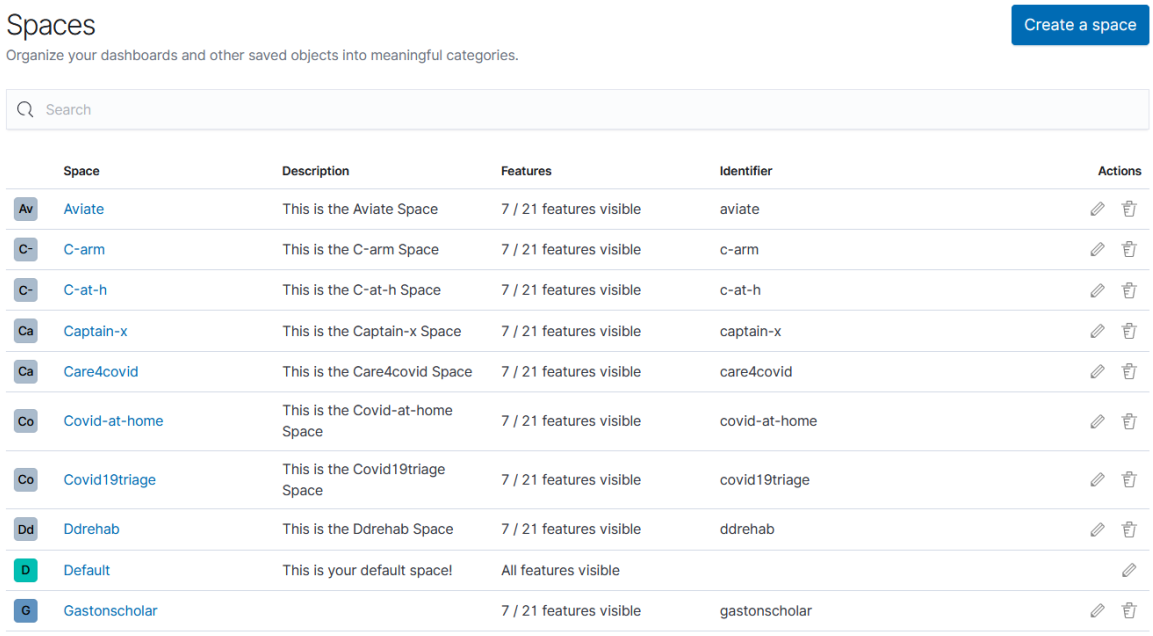

FIGURE 16: KIBANA SPACE DESCRIPTION

Kibana creates a default space, where users with the kibana_admin built-in role can grant access to all Kibana features in all spaces.

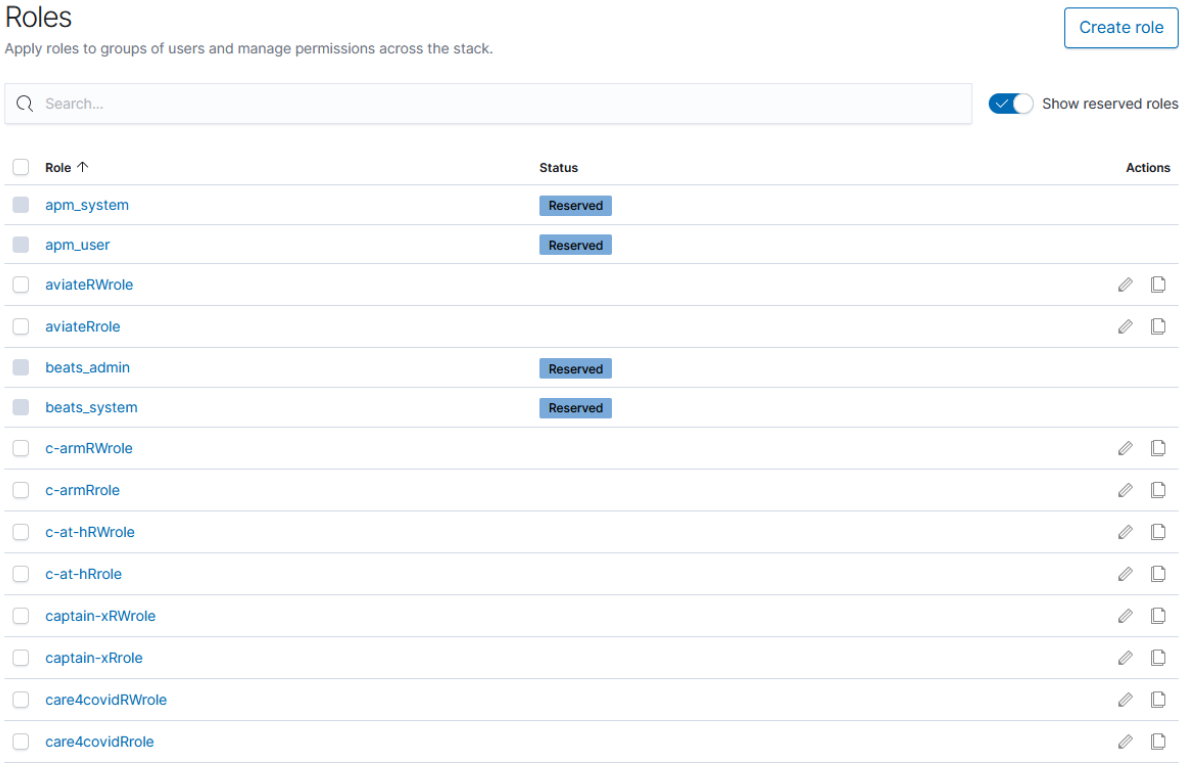

FIGURE 17: KIBANA SPACE ROLE MANAGEMENT

A subset of the roles created for the COVID-X Sandbox, showing the differentiation between the RWrole and the Rrole for each third party.

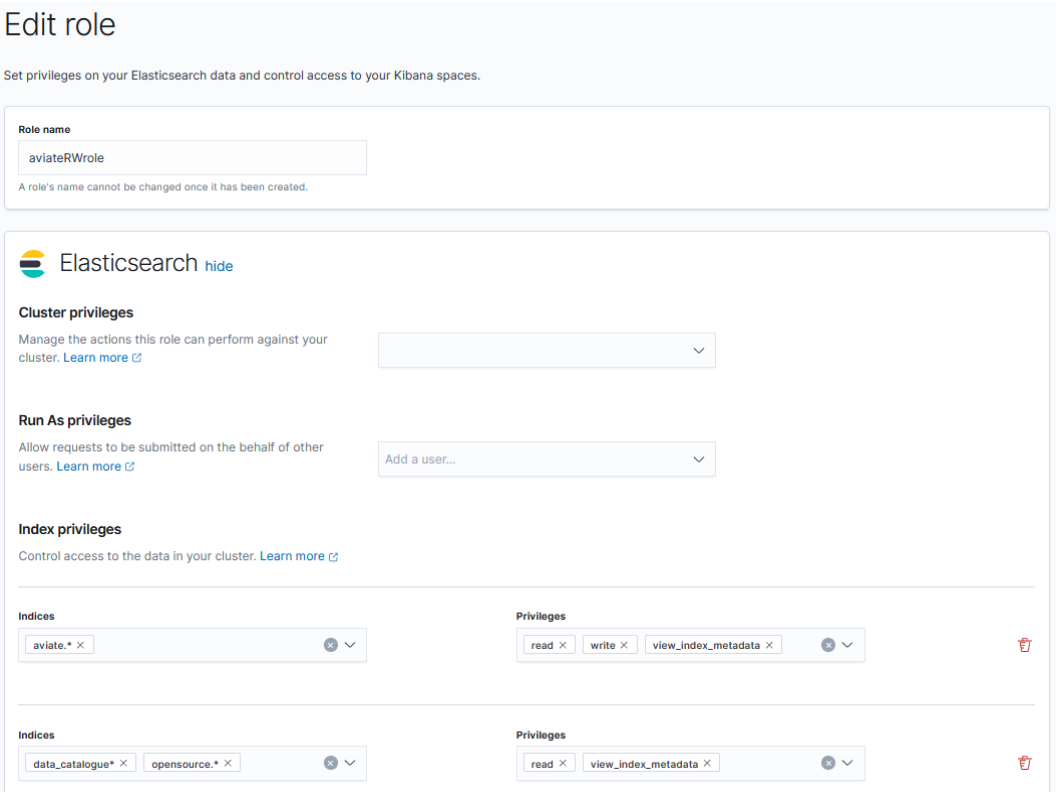

FIGURE 18: KIBANA SPACE WRITE ROLE CONFIGURATION

Example of a RWrole acting as the admin for a specific organizations' indices. Read-only access to the data_catalogue and opensource indices is shown.

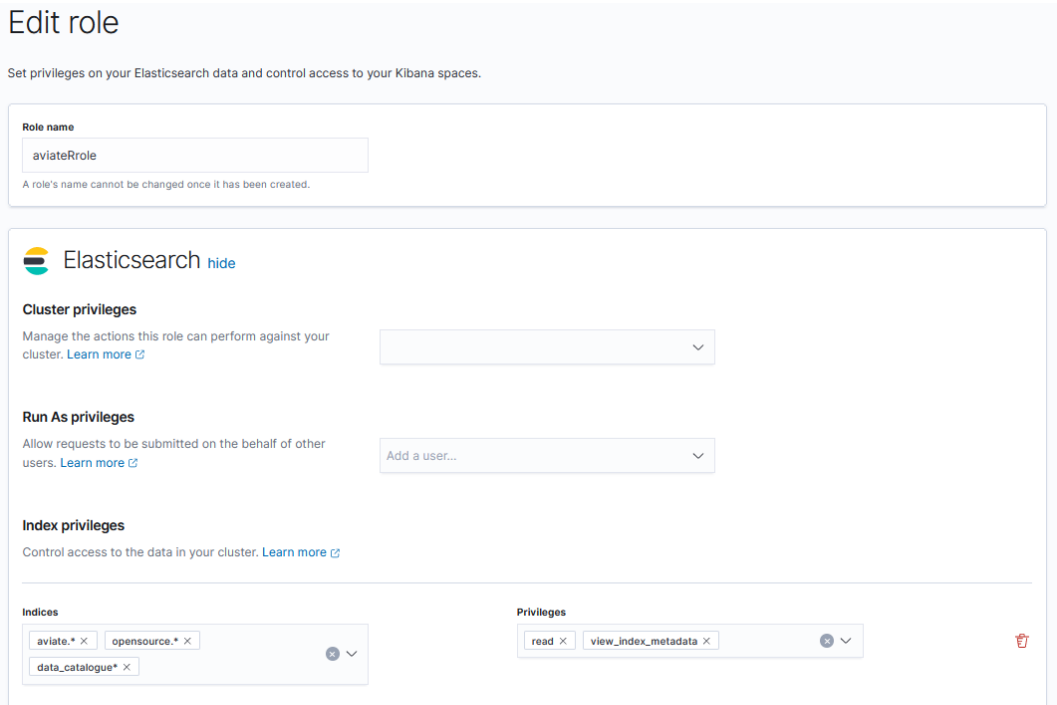

FIGURE 19: KIBANA SPACE READ ROLE CONFIGURATION

Example of a Rrole (read-only role). Read-only access to the data_catalogue and opensource indices is shown.

With regards to encryption: In the COVID-X Sandbox, we capitalize on the XPack extension of the ELK Stack to create three levels of encryption: in the application layer communication with Kong (HTTPs), in the transport layer communication between elastic nodes (TLS), in the application layer communication for both ELasticsearch and Kibana (HTTPs).

For the TLS encryption to work, X.509 certificates are required. These certificates ensure both the sender's identity and the message's security itself.For the communication between nodes to be truly secure, the certificates must be signed by a publicly trusted issuer Certificate Authority (CA). Self-signed certificates raise users' awareness of the risk they assume when deciding to resume the requested communication. For the second release of the COVID-X Sandbox, we use *Let's Encrypt* certificates. *Let's Encrypt* is a non-profit Certificate Authority (CA) run by Internet Security Research Group, that provides X.509 certificates for TLS encryption at no charge. The certificates are trusted by most common browsers/tools and enable HTTPS on a website/service. Based on this approach, new nodes need to have *Let's Encrypt* certificates to join the ELK cluster. Therefore, secure communication between the ELK cluster and external users is achieved, since unauthorized nodes that do not have the required certificates cannot intercept the ELK cluster.

With regards to data availability: The centralized instance of the COVID-X Sandbox is deployed on five virtual servers instantiated on Hetzner public cloud. It incorporates a load balancer -Kong- used to redirect requests to all nodes established in the cluster, in a way that distributes the load appropriately, i.e., by considering available resources of each node, their computational power and status (up, down). This property is beneficial to the Sandbox, as it allows high availability and increased performance when handling incoming requests.

3.4 Sandbox Interfaces for Third Parties Integration

The new endpoints released since D2.2 [2] have achieved an easy to use, unsupervised way of interacting with the sandbox. The new endpoints cover several functionalities not previously offered, but needed for the technical team to carry out constant supervision. Overall, the complete set of endpoints offered by the sandbox are gathered below.

Method	Endpoint	Description
POST	/api/ingest_stream/<org>/<index>	targets logstash service ingesting in a specific index
POST	/api/ingest_file/<org>/<index>	upload a file into the logstash target folder. File upload must be a multipart request with "file" key
GET/POST	/api/search_index/catalogue	returns data catalogue information stored in elastic
GET/POST	/api/search_index/<org>/<index>	elastic DSL query targeting specific index
POST	/api/delete_on_index/<org>/<index>	elastic DSL query for deletion regarding a specific index
POST	/api/delete_on_team/<org>	DSL query for deletion regarding all owned indices
GET/POST	/api/scroll	query with pagination

Method	Endpoint	Description
GET (browser)	/	kibana GUI exposed on the root path
-	/client/search_index	endpoint for PL clients

TABLE 3: COVID-X SANDBOX THIRD-PARTY ENDPOINTS

Basically, the new endpoints comprise results pagination, programming language clients support, delete on team indices and programming language clients. The following points provide a detailed description of each endpoint:

- **Pagination**. Elastic provides two ways of pagination capabilities: scroll and pit. Scroll is the traditional manner and is the chosen due to the familiarity the SMEs may have over the pit endpoint. Thus, the request consists of two stages: creating the scroll ID with the first search query and using it with the scroll endpoint as desired.
- **PL support**. Software designs usually depend on connectors to databases to transmit the information. As the ELK stack is the core of the sandbox and offers elastic, the API should be capable of interacting with PL libraries. The sandbox provides access to the search capabilities using these libraries.
- **Delete on team**. Tweak over the normal index deletion provided in the previous release. It allows deleting certain documents, filtered by a DSL query, across all the indices owned by a specific team. It is useful when the data has some shared keys across the indices not to repeat redundant delete queries.

3.5 Federated Learning and Validation Services

FL is a data driven innovative technique capable of deploying machine learning models in the several nodes where data is stored. In its architecture, there are several clients interacting directly with data and one server responsible for requesting information from the clients and combining it to advance the learning process. All those interactions occur remotely, and the client-server communications just involve: sending to the servers the results of each individual training step produced in the clients ;and transferring to the clients the new state of the model where the next training step starts. FL moves the computation to the data, and not the opposite; hence it is essential in projects related to both data and health to comply with data protection policies.

Nowadays, there are several FL frameworks with specific functionalities leading to different possibilities in terms of settings. Not using a framework is not a possibility, since creating this service from scratch is a complex time-consuming work not affordable in the context of COVID-X.

3.5.1 Architecture

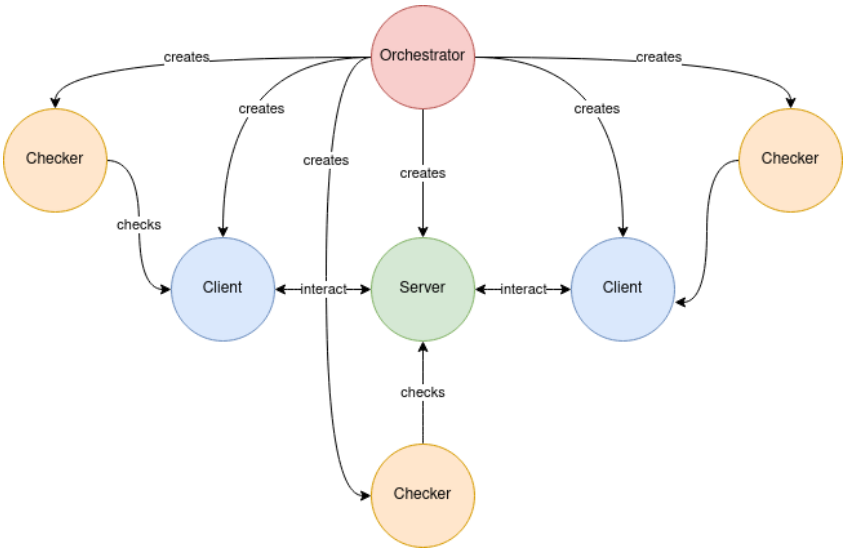

FIGURE 20: FEDERATED LEARNING SERVICE OVERALL ARCHITECTURE

Flower

Flower is an opensource simple scalable model agnostic FL framework that handles communications via RPC (Remote Procedure Call) protocol. RPC has a nice interaction with docker containers as it doesn't need to publish the ports of the worker (client) nodes. Flower is simple to use and supports numerous frameworks or even custom python code, hence it is also pythonic and thus delivers an easy-to-use API.

One of the most important requirements for the sandbox is that the FL service is not intended for the technical team, but for the SMEs. That is why it is not enough to produce model agnostic API, but also a clean and easy to use interface. This way, even if the SMEs are not proficient in producing a FL model, the complexity is hidden in the background and the model python scripts are straightforward.

Flower needs two different scripts with completely different roles: a central server where individual models are combined and a client training on the node data at each iteration;.and implementation of the server code combining models and client code regarding each of the training steps. Moreover, Flower just introduces approximately 20 lines of code adding up both client and server codes, in its basic usage example.

Orchestrator

The name given to the container handling the whole FL process, from receiving the rest calls to producing the desired containers and checking the status of each task, is the term orchestrator. The orchestrator is a Python class wrapped over a Python flask application just to expose some endpoints of interest. The next paragraphs detail the orchestrator design from API endpoints to the end of a FL task.

Flask has already been used in this project and hence commented in D2.2 [2]. Flask is a library that creates a server capable of receiving http requests in an easy manner. Flask is responsible for listening to a defined port and telling the orchestrator the properties of each new FL task received.

Once flask receives a request, it produces a unique ID and path to where the results should be stored and passes it to the orchestrator. The request must contain a list of named nodes to which the clients need to be deployed, a server.py script and a client.py script. All this information is gathered in a task object and appended to the queue of tasks the orchestrator contains. The result of the request is the unique ID of the task that the program returns to the flask's client.

The orchestrator checks the length of the queue of tasks every time there is a change of its state, so the tasks are launched sequentially, thus executing several tasks parallelly is avoided. This way there is only room for one FL task at a time and the resources necessary for this functionality can be assigned entirely to each task.

Manager

Apart from the orchestrator class, there are several bash scripts playing the role of helpers to this object. The most important one is the manager, who is responsible for accepting several parameters and deploying a new container in any remote or local machine.

The manager acts in different ways depending on the arguments passed to it. These different roles work as follows:

- Deploy mode: targets a machine's IP or localhost and deploys either a client or a server. It copies the desired scripts into the remote machine (if needed) and executes them via SSH or locally.
- Clean mode: targets a machine's IP or localhost and cleans all the containers and images with server or client names.

Closely related to the manager, there are several bash scripts responsible for each possible task assigned to the manager. In any case, if the target is a remote host, the desired script is copied to the target and executed through ssh; or if the target is localhost, the script is executed directly. These are the scripts referred in the two previous points.

Obviously, the container in charge of running the orchestrator and manager has two important requirements:

- Its public key must be known in advance by all the available remote hosts. Hence, it needs to be static and accessible.
- The container needs to allow local deployment because all three, orchestrator, server and client could run on the same host or cluster. In order to provide such a requirement, the orchestrator must have access to its host's docker service. To do so, the image is built from a Docker image and maps its own /var/lib/docker to the same path of the host.

Checker

The last component in terms of architectural definition is the script responsible for checking the behaviour of each piece involved in a specific task of FL. There is one checker per client targeting either local or remote containers, and another checker verifying the status of the server.

The checker communicates with the docker service and checks whether the containers are up and running. In the case the container is not visible for several checks, the checker triggers the end_task mechanism on the orchestrator.

3.5.2 Workflow

The general workflow has already been described in the architecture at a high level. The overall process of defining and creating a process from API request to the end of the process has several components that, as discussed, work together to deliver the FL service. This section covers a more detailed explanation of the whole process of creating a FL task.

The API request arrives into the orchestrator's flask with a list of hosts where the service should run and both server.py and client.py files. Afterwards, flask creates a unique ID and a path for that job, then passes the information to the orchestrator. The orchestrator creates a new task object, responsible for storing the information of a job and adding it to the queue. The queue is a FIFO structure granting the correct execution of tasks in order.

When a job is selected for running, the orchestrator changes its state into running mode and waits for the current task to terminate (to avoid having several tasks in parallel). The task object is popped out from the queue and the information creates an execution chain in terms of subprocesses as follows[12]:

- The manager is called to deploy the server. Normally it is done via local bash script but prepared to work remotely. The server is created automatically via docker build and docker run.
- The manager is called to deploy each client (one subprocess for each client). The client is created automatically via docker build and docker run.
- The manager creates one checker for each entity involved in the job. Each of the checkers verifies the client/server is running, and no errors have occurred.

When the job finishes, the orchestrator receives a finish signal from the server, producing a clean order in the manager. Then, the manager cleans all the environment (each machine involved in the FL process) in terms of docker containers and images. Moreover, the orchestrator sends a terminate signal to its children to end the checker subprocesses. When an error occurs, the orchestrator receives a finish signal with information about the error and the same process of cleaning happens.

In any case, logs are stored locally in the directory to which the task object points. The folder contains information about the client and server scripts accompanied by the logs produced in each node and a summary of the process' results.

[12] If the target is a remote machine, the manager first runs another script that prepares the folder structure

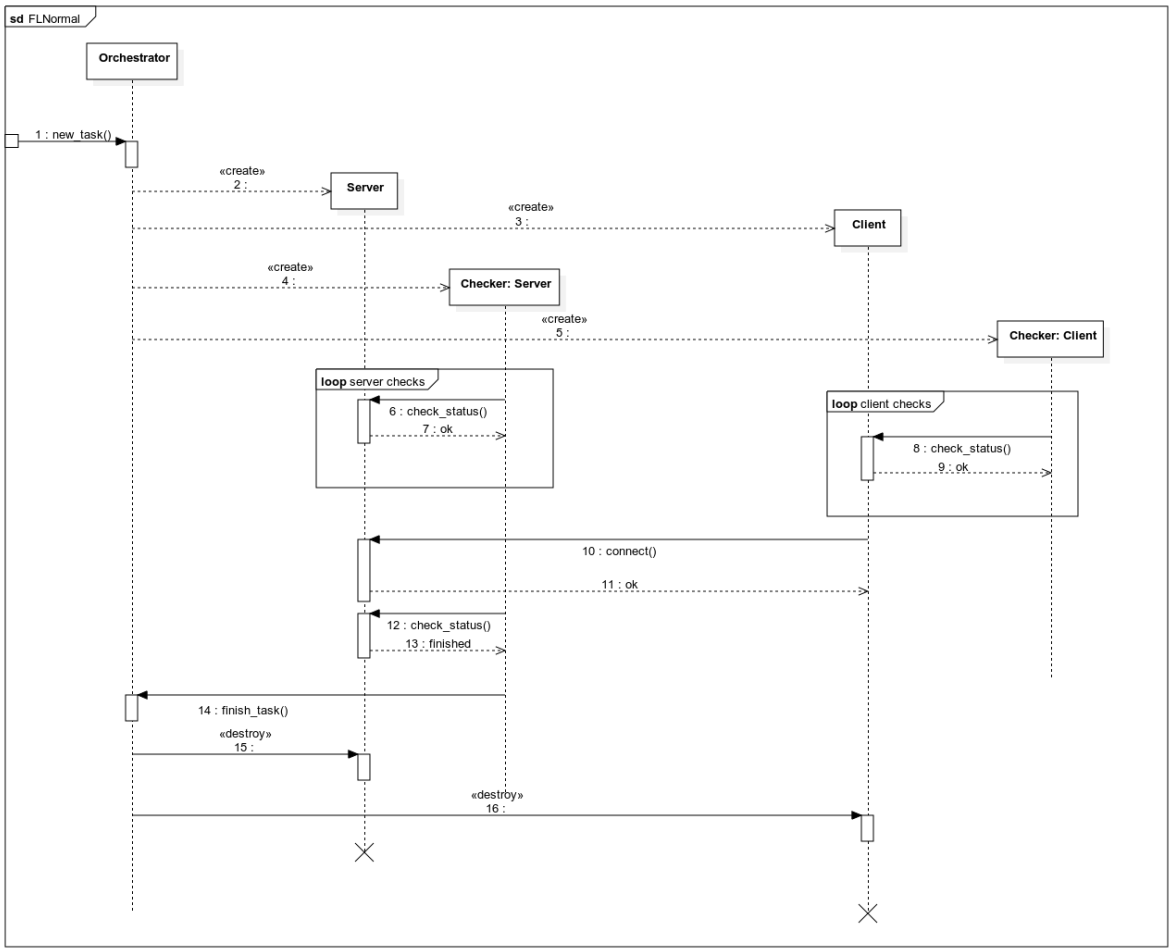

FIGURE 21: FEDERATED LEARNING SERVICE - SEQUENCE DIAGRAM (NORMAL EXECUTION)

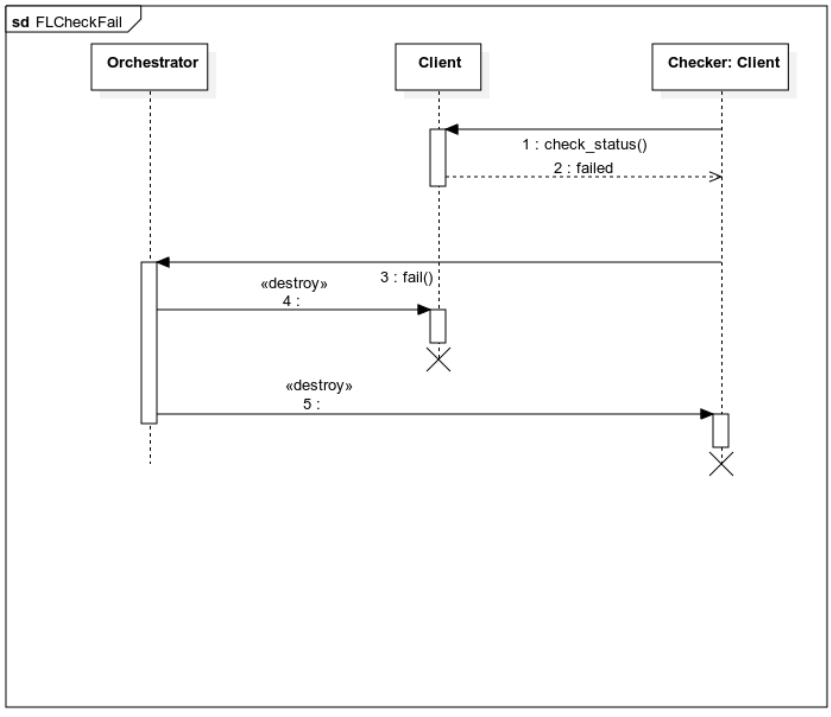

FIGURE 22: FEDERATED LEARNING SERVICE - SEQUENCE DIAGRAM (HEALTH CHECKS IN CASE OF ERRORS)

3.5.3 Security

The FL service handles security in a similar manner to how the data ingestion endpoints work. ELK stack is responsible for managing the process of authentication both users and applications. Elastic supports application privileges, useful in defining the roles that an application can support. In other word, the role is linked to an application (federated learning) and can be used by any user linked with this role.

The process a request undertakes until authentication is as follows. First, the orchestrator catches the request's authorization header. Remember all requests from the end users have specified their credentials and the clinical partner for which they are creating a training instance. Then, the orchestrator checks the credentials against elastic with the has_privileges endpoint and, if granted access, allows creating the job.

The same procedure applies to all endpoints related to this functionality so the security aspects are well covered.

3.5.4 API

The endpoints offered by the sandbox in terms of the FL service are described in the following table.

Method	Endpoint	Description
POST	/api/fl/create_task/<org>	Creates a FL job that is added to the queue. Returns an ID.
GET	/api/fl/status/<ID>	Returns the status of the FL task. If finished, returns the results.
DELETE	/api/fl/cancel_task/<ID>	Cancels the task if it has not run yet.

TABLE 4: COVID-X SANDBOX - FEDERATED LEARNING SERVICES ENDPOINTS

FL tasks are created on demand with the create_task endpoint. This task triggers the launch mechanism if there are no queued tasks and returns the task ID so that the user can verify the status at any moment. When the job is finished, the results are showed for the user to check.

4 Sandbox Deployment

This chapter presents an overview of the final version of Sandbox deployments to support both COVID-X Clinical partners needs and their legal/operational constraints, as described in D2.1 [1], as well as the needs of the solutions included in COVID-X project during OC1 process. The first part of this section provides details about the centralized COVID-X Sandbox deployed in a cloud-based environment. The second section describes all the aspects related to the deployments established on clinical partners' premises regarding the process and the infrastructure used for Sandbox purposes.

4.1 Cloud-based Deployment - Final Release

The cloud-based (centralized) version of COVID-X Sandbox is deployed on a set of five virtual machines on Hetzner public cloud, as shown in the figure below. All the core COVID-X Sandbox services that illustrate the data ingestion, data storage and data visualization capabilities have been deployed and configured in this infrastructure. All data stored and processed in the cloud-based deployment are fully anonymized, follow the GDPR guidelines. For this purpose, Hetzner provides the option to create a Data Protection Agreement (DPA) following Article 28 of the General Data Protection Regulation (GDPR) by filling a form in the Hetzner dashboard. In addition to data coming from clinical and OC1 data providers, selected publicly available datasets are included in the cloud-based deployment of COVID-X Sandbox.

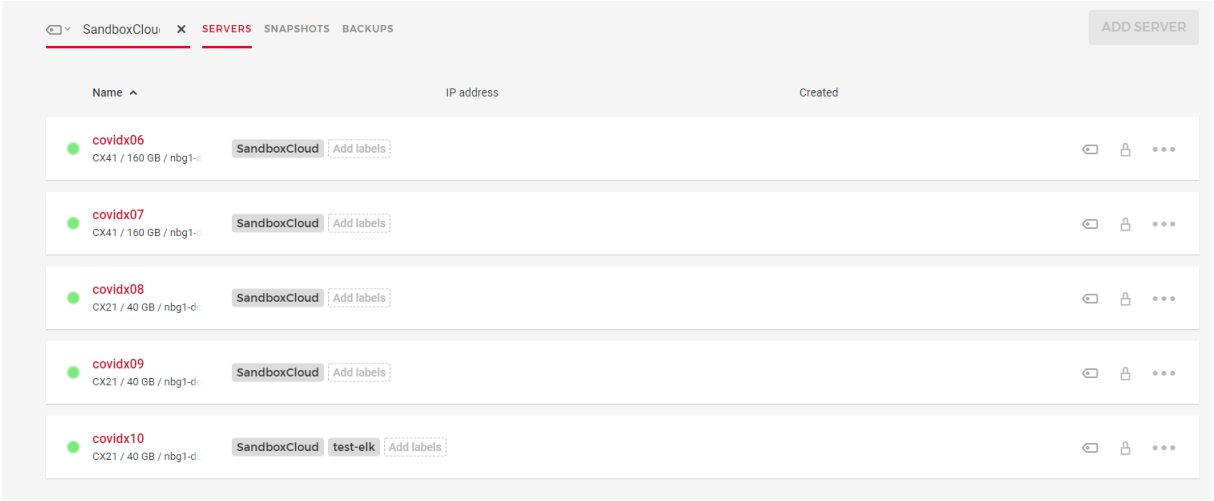

FIGURE 23: COVID-X SANDBOX CLOUD-BASED VERSION (HETZNER SETUP OVERVIEW)

Hetzner virtual machines, used for COVID-X Sandbox purposes, are attached to a private network that ensures secure internal communication. Also, all the machines are assigned to a public IP address, enabling external communication. External communication is protected by dedicated firewalls running on each machine and security policies enabled by the cloud provider, allowing access only to authorized users.

The next table lists all computational resources to setup and deploy COVID-X Sandbox services in the cloud-based environment. Currently these resources cover mostly the needs of COVID-X Sandbox OC1

solutions, but it is feasible to extend them and rescale the whole infrastructure to achieve properly any further needs (e.g. OC2 needs, increased functionality of existing solutions).

Virtual Server	vCPU	Memory	Storage
covidx06	4	16 GB	160 GB
covidx07	4	16 GB	160 GB
covidx08	2	4 GB	40 GB
covidx09	2	4 GB	40 GB
covidx10	2	4 GB	40 GB

TABLE 5: COVID-X SANDBOX CLOUD-BASED VERSION (COMPUTATIONAL RESOURCES)

COVID-X Sandbox services are deployed in a distributed way running in Docker containers. Docker daemon has been installed on all the virtual machines, configured to run in swarm cluster mode. This mode enables multi-host Docker networking, necessary for cross-server communication between containers across COVID-X Sandbox cluster. A dedicated overlay Docker network has been created for this purpose, enforcing encrypted traffic between the Docker containers.

```
[cvadmin01@covidx06 ~]$ docker node ls
ID                            HOSTNAME   STATUS    AVAILABILITY   MANAGER STATUS   ENGINE
y6skogz5yk45pk9pp6gg062z4 *   covidx06   Ready     Active         Leader           20.10.6
chuav8x0fkofjzmqag5sw1nuc     covidx07   Ready     Active                          20.10.6
h5oz74zkggnogcl1s7bfjrcm9     covidx08   Ready     Active                          20.10.6
sp5j4gs16nwqsln47v9lls93g     covidx09   Ready     Active                          20.10.6
w42qpy5ccjefueknz61s849e2     covidx10   Ready     Active                          20.10.4
```

FIGURE 24: COVID-X SANDBOX CLOUD-BASED VERSION (DOCKER NODE OVERVIEW)

Administrator users from COVID-X technical partners are responsible for the setup and maintenance of COVID-X Sandbox cloud-based deployment, and the development and configuration of its core services. The deployment of this environment follows the workflow of the CI/CD process initially introduced in D2.2, and further discussed in section 2 of the current document. This workflow uses Docker images that have successfully passed the necessary unit and integration tests and have been pushed into the project's private repository. Docker containers are deployed using Docker compose files and wrapper shell scripts. The Table below presents the deployed containers comprising the Sandbox services.

Container Name	Application	Sandbox service
sandbox-elasticsearch-master	Elasticsearch master node & X-Pack	Data Management & Harmonization services, Security services
sandbox-es-node1	Elasticsearch data node & X-Pack	Data Management & Harmonization services, Security services
sandbox-es-node2	Elasticsearch data node & X-Pack	Data Management & Harmonization services, Security services
sandbox-es-node3	Elasticsearch data node & X-Pack	Data Management & Harmonization services, Security services

Container Name	Application	Sandbox service
sandbox-kibana	Kibana & X-Pack	Visualization services, Security services
sandbox-logstash	Logstash	Data Management & Harmonization services
sandbox-apigw	Kong	Sandbox interfaces
sandbox-ingestfile	Python Flask	Sandbox interfaces
sandbox-datalake	MongoDB	Data Management & Harmonization services

TABLE 6: COVID-X SANDBOX CLOUD-BASED VERSION (DEPLOYED SERVICES)

4.2 Local Deployments on Clinical Partners Premises

Apart from cloud-based deployment of COVID-X Sandbox, some local deployments have been set up on the premises of data providers/clinical partners. The reason behind these deployments was to avoid distribution issues regarding closed anonymized data access and to ensure GDPR and national regulations compliance. Additionally, it also addresses the need for Data Controllers to keep the control of the closed datasets.

To go through a local deployment, data providers offer an infrastructure on their premises, able to host a fully functional instance of COVID-X Sandbox. Typically, a multi-core machine with 64GB of RAM should be available with SSD Drives. The disk space must be a multiple of the data volume to be stored. Multiple hosts of 3x64GB RAM, or 3x32GB RAM are an ideal setup, or 3x16GB RAM machines are significantly better and allow for a Highly Available (HA) setup. If necessary, the whole setup can be easily scaled to match the needs of the system. Remote access to this infrastructure through SSH connection is necessary to allow accessibility to COVID-X Sandbox services, preferably over a secured VPN tunnel. Administration privileges are also required.

4.2.1 ICH

ICH provided a virtual machine (VM) hosted on the Google Cloud Platform, which has been legally certified to host their data. Access has been provided only to certified users with the appropriate permissions to configure the prerequisites of the COVID-X Sandbox and adjust its deployment to the underlying infrastructure. The available hardware infrastructure includes a multi-core machine with 64 GB of RAM and 100 GBs of disk space.

5 Sandbox User Guide

This section provides all the necessary information that administrators need to install and configure the final release of COVID-X Sandbox in an operational environment consisting of a set of physical or virtual machines. Additionally, it provides a guide to end users on how to access and interact with the services of COVID-X Sandbox to implement and support their own use cases.

5.1 Sandbox Installation and Configuration

The final release of COVID-X Sandbox is built mostly on the same basis as the first release. It focuses mostly on updating the existing services, but from infrastructure and setup perspective follows the same guidelines as the first release. COVID-X Sandbox deployment requires at minimum one virtual or physical machine with 64GB of RAM, equipped with all the necessary SSD drives. It is preferable for Sandbox deployment, the disk space to be multiple of the data volume to host. Also, it is possible for COVID-X Sandbox services to be distributed into multiple hosts, as long as it is possible to communicate to each other.

D2.1 – First Sandbox Implementation and Services Provision provides a very detailed guide, on how COVID-X Sandbox can be installed and configured in a set of a virtual or physical machines to provide an operating environment for third parties that need to use its services. In this guide, the technologies and corresponding versions to be installed are defined, the proper Gitlab repositories and instructions about the CI/CD stack are listed also with information on how they can be used for setting up an operating instance of COVID-X Sandbox and its services.

5.2 Sandbox Services Usage Guide

5.2.1 OpenVPN

As presented in Section 2.2.2, the COVID-X Sandbox is protected by a dedicated firewall, and access to its services is granted through a private VPN tunnel. Sandbox hosts an OpenVPN server, and its clients must use an OpenVPN client to establish a secure VPN tunnel. OpenVPN Clients are supported by most operating system platforms, such as Windows, Linux, macOS, Android, and iOS. The appropriate software application can be downloaded and installed on the respective client system by the OpenVPN community repository[13]. The installation process is simple and can be completed in a few standard steps. Figure 25 presents the steps for installing the OpenVPN client on a Windows system.

[13] https://openvpn.net/vpn-client/

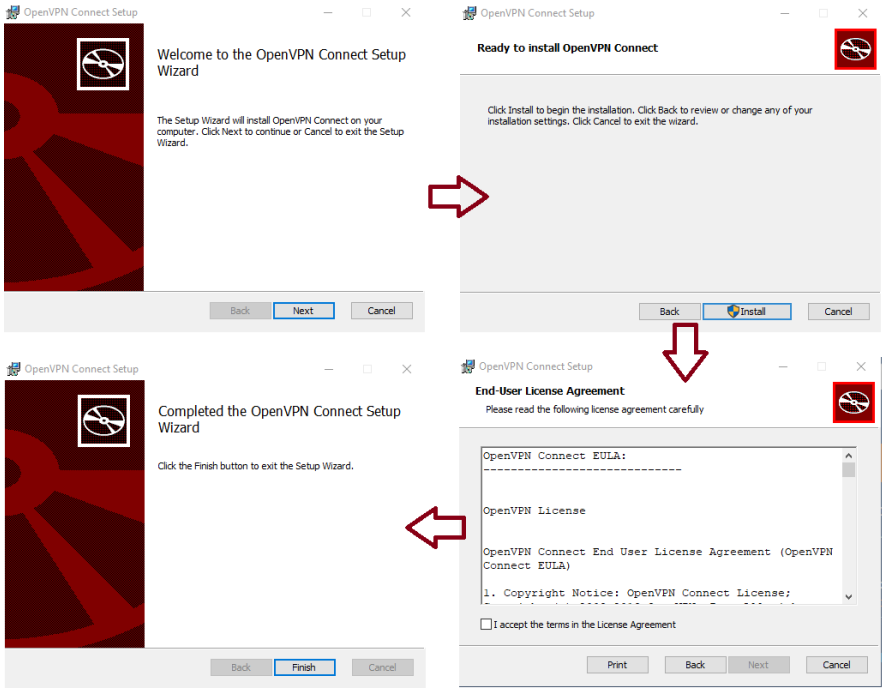

FIGURE 25: OPENVPN CLIENT INSTALLATION ON MICROSOFT WINDOWS

Having installed the OpenVPN client, the user needs to get an OpenVPN configuration file and credentials for the corresponding account. The Sandbox administrators provide this information through a secure and trusted channel. The OpenVPN configuration file contains information about the VPN tunnel, including the public URL of the Sandbox VPN server, the client certificate used for encrypting the traffic, and the IP routes that must be installed on the client's system to redirect the network packets destined for the Sandbox services through the private VPN tunnel. The configuration file must then be imported to the OpenVPN client, creating a new connection, as depicted in the following Figure.

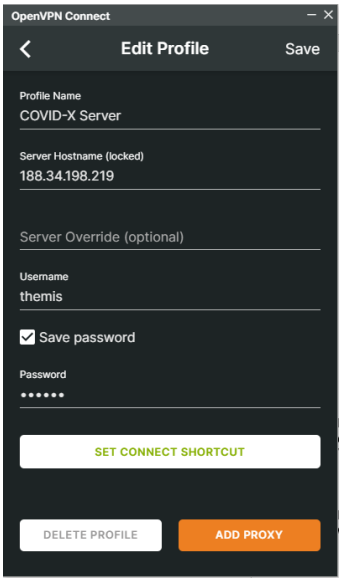

FIGURE 26: NEW OPENVPN CONNECTION

The final step is to activate the new connection, establishing the private VPN tunnel:

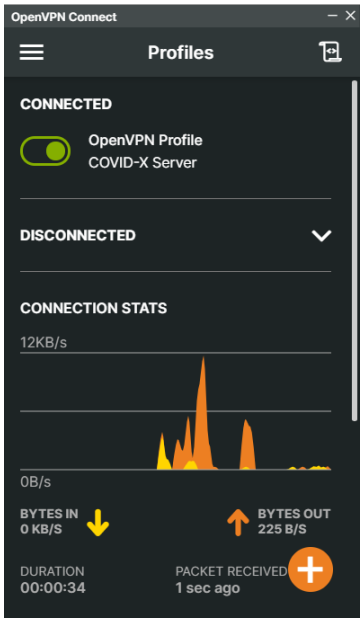

FIGURE 27: ESTABLISHED OPENVPN CONNECTION

5.2.2 Kibana

Kibana, as described in previous sections, is used for end users to visualize data they can access. The main URL is used to access Kibana services is the following: https://sandbox.covidx.rid-intrasoft.eu . After providing credentials for logging in, Kibana space selector (https://sandbox.covidx.rid-intrasoft.eu/spaces/space_selector) appears ahowing the spaces the logged user can access.

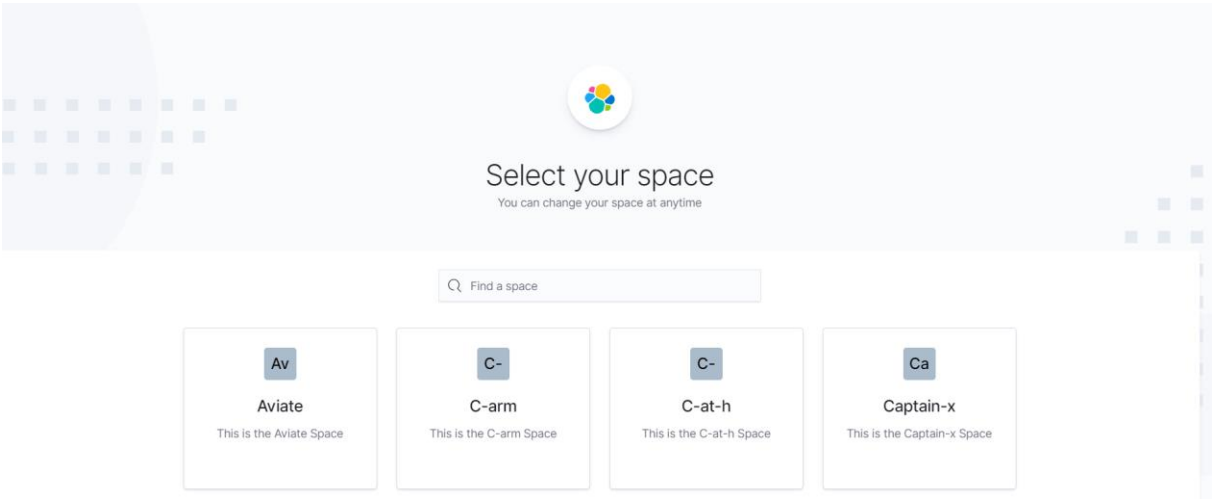

FIGURE 28: KIBANA SPACE SELECTION (COVID-X ADMIN USER)

By selecting a specific space, end users can create their own visualizations (dashboards, maps, etc.).

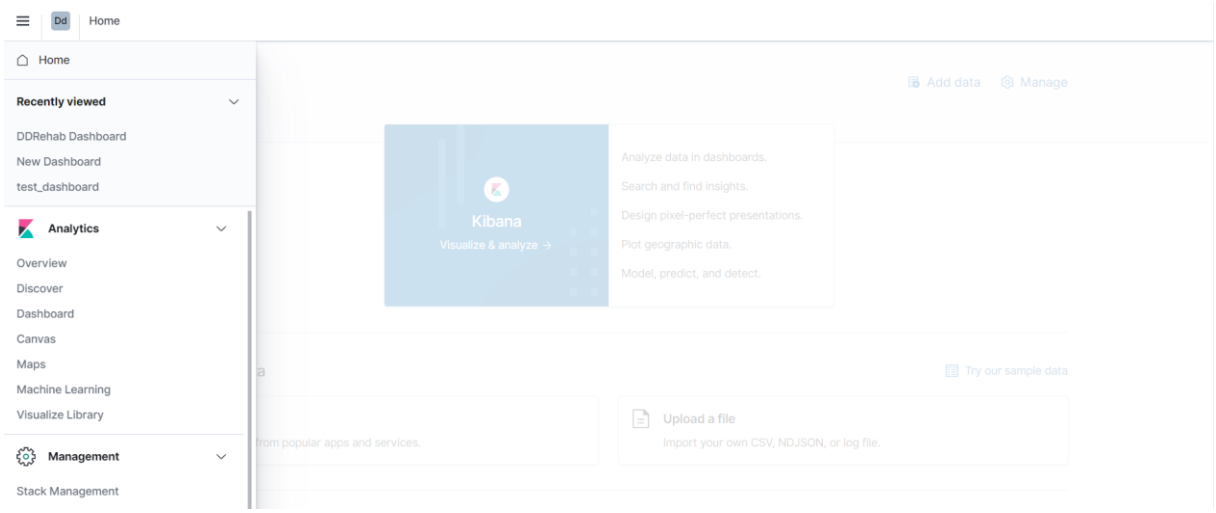

FIGURE 29: KIBANA VISUALIZATION HOME

If Dashboard option is selected, users can either view or modify existing dashboards, or create new ones.

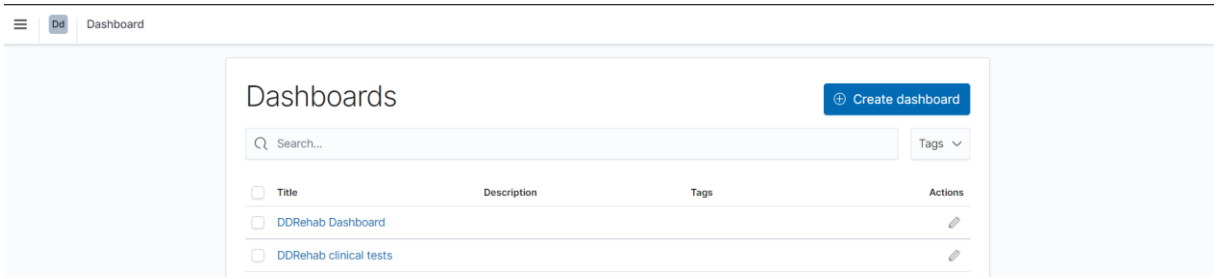

FIGURE 30: KIBANA DASHBOARD HOME

By selecting a specific dashboard, several visualizations can be created on top of indexed data existing on Elasticsearch. Kibana offers a wide variety of visualization modes (https://www.elastic.co/guide/en/kibana/current/get-started.html) and provides flexibility on how these visualizations can be organized.

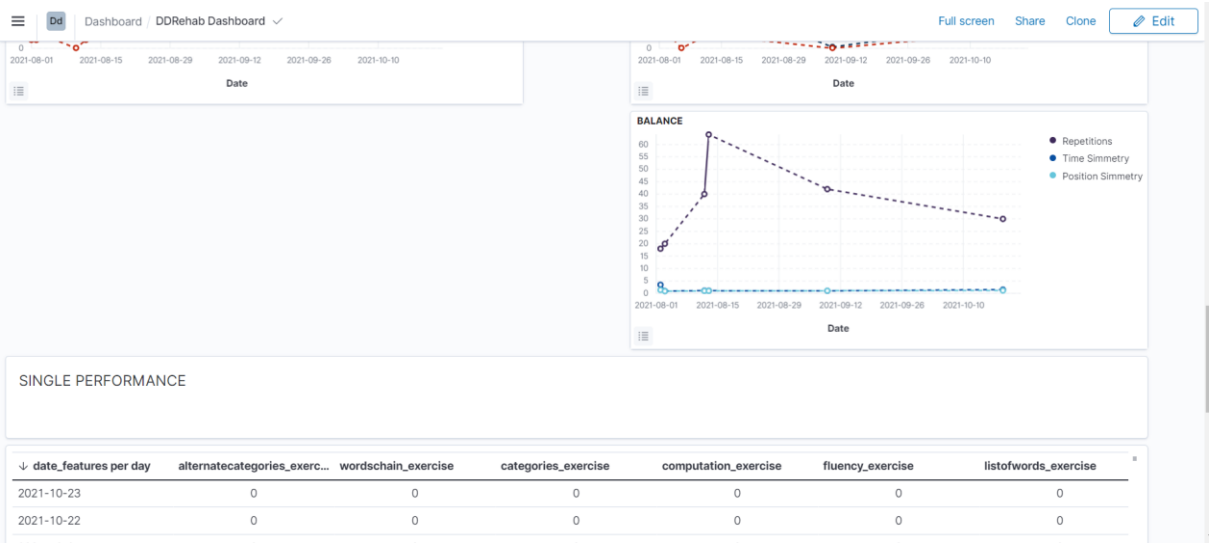

FIGURE 31: KIBANA DASHBOARD EXAMPLE

5.2.3 Elasticsearch

Elasticsearch can be easily accessed by end users through defined RESTful endpoints for performing search queries on top of indexed data and deleting indexing if necessary. All these endpoints are described in detail both in the D2.2 [2] and the current document which gives a full overview of all the actions an end user can do on top of indexed data store in Elasticsearch. RESTful endpoints can be accessed through API calls using either cURL commands or tools to create REST calls (e.g. Postman). Calling these APIs requires users to have valid credentials. Below, sample commands can be found that show the interaction of users with the Elasticsearch:

- Query specific index under an organization

```
curl --request GET 'https://<username>:<password>@sandbox.covidx.rid-intrasoft.eu/api/search_index/<org>/<index>
```

- Delete specific index under an organization

```
curl --location --request POST 'https://<username>:<password>@sandbox.covidx.rid-intrasoft.eu/api/delete_on_index/<org>/<index>' --header 'Content-Type: application/json’ --data-raw '{“query”: {“match_all”: {}}}'
```

D2.4 [3] provides a more detailed guide with complete examples on how these APIs are used in terms of end-user interaction to COVID-X Sandbox.

5.2.4 Logstash

Similar to Elasticsearch, end users access Logstash through REST API calls, cURL commands or tools to create REST calls (e.g. Postman). These APIs are used only for sending data from clinical data provider to COVID-X Sandbox for further processing, harmonizing and indexing to Elasticsearch. Below, sample commands can be found that show the interaction of users with the Logstash to provide data for ingestion:

- Provide stream for data ingestion related to a specific index under an organization

```
curl --location --request POST 'https://<username>:<password>@sandbox.covidx.rid-intrasoft.eu/api/ingest_stream/<org>/<index> --header 'Content-Type: application/json' --data-raw '{"foo": "bar"}'
```

- Provide a file for data ingestion related to a specific index under an organization

```
curl --location --request POST 'https://<username>:<password>@sandbox.covidx.rid-intrasoft.eu/api/ingest_file/<org>/<index>' --form 'file=@"/home/user/desktop/file1.csv"'
```

D2.4 [3] provides a more detailed guide with complete examples on how these APIs are used in terms of end-user interaction to COVID-X Sandbox.

6 Conclusions

This document presented and described further the final release of COVID-X Sandbox platform, focusing mostly on the architectural and functional updates implemented, compared to the first COVID-X Sandbox released on month 6 of COVID-X project and was the primal subject of D2.2 [2]. It described all the technical fixes or updates in terms of the CI/CD stack used to deploy COVID-X Sandbox services in an operational environment. From architectural perspective, the introduction of Federated Learning services, as well as the definition of a common Semantic Data Model were the two main additions in this final release, compared to initial one.

The final release of COVID-X Sandbox is still compliant with all the best practises of CI/CD process, as they were presented in D2.1 [1], and the development of their services follow the Service-Oriented Architecture approach. Additionally, improvements in security services have been presented, to ensure access to COVID-X Sandbox services and availability of the data stored only to users with authorized permissions.

Finally, this document shows how COVID-X Sandbox was deployed both in a cloud-based and in local environments, to illustrate the needs of use case scenarios covered during COVID-X projects. It also includes a guide describing how users can access COVID-X Sandbox services.

7 References

[1] COVID-X Consortium, “D2.1 – Sandbox Design & Datalake Creation and Ingestion”, 2021
[2] COVID-X Consortium, “D2.2 - First Sandbox Implementation and Services Provision”, 2021
[3] COVID-X Consortium, “D2.4 - First Report on interconnection of third parties”, 2021

8 Appendix – Updated COVID-X Semantic Data Model

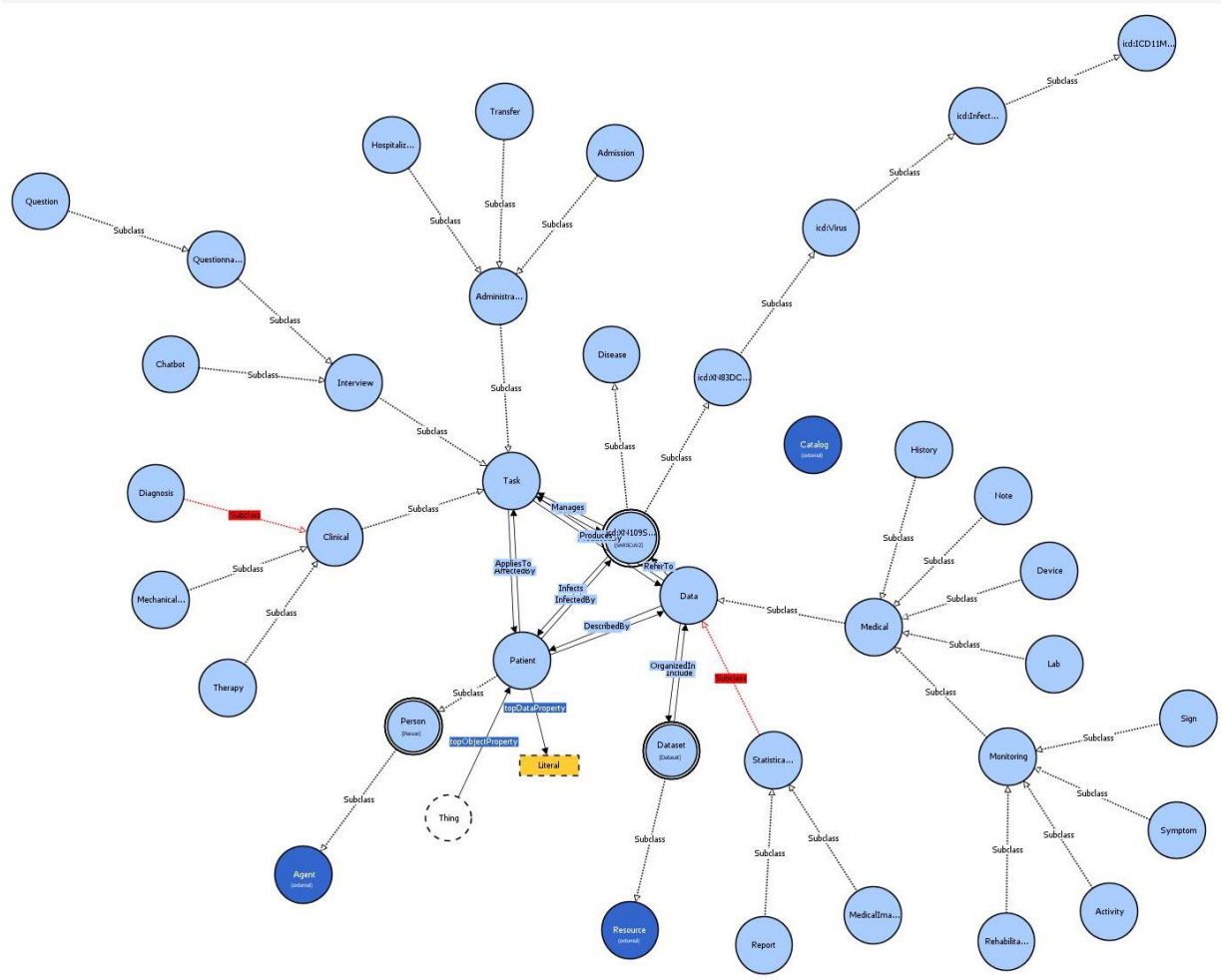

FIGURE 32: COVID-X SEMANTIC DATA MODEL - FULL ONTOLOGY