COVID EXPONENTIAL PROGRAMME

GRANT AGREEMENT ID: 101016065

D2.2 – First Sandbox Implementation and Services Provision

Revision: v.1.0

Work Package	WP2
Due date	30/04/2021
Submission date	30/04/2021
Deliverable lead	INTRASOFT
Version	1.0
Authors	Themistoklis Anagnostopoulos, Serafeim Tsironis, Sofia Tsekeridou (INTRASOFT) Alexandros Karatzos, Despina Gkatzioura (8Bells) Borja Aroyo, Silvia Uribe, Gustavo Hernandez (UPM)
Reviewers	Antonio Damasceno (F6S)

Abstract	This deliverable presents the first release of the COVID-X Sandbox and

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

	describes the services offered, including data ingestion, harmonization, visualization, analytics, recommendation, API connectivity and management
Keywords	Sandbox, data management, data harmonization, data integration, visualization, elasticsearch

DOCUMENT REVISION HISTORY

Version	Date	Description of change	List of contributors
V0.1	08/02/2021	Initial ToC	INTRA
V0.2	15/02/2021	Added chapter 2 – CI/CD Stack	INTRA
V0.3	10/04/2021	Added chapter 3 – Sandbox integrated services	INTRA, UPM, 8BELLS
V0.4	20/04/2021	Added chapter 4 – Description of the Sandbox deployment	INTRA, 8BELLS
V0.5	25/04/2021	Added chapter 5 – Sandbox User Guide	INTRA, UPM, 8BELLS
V0.6	27/04/2021	Updates in all chapters, introduction and conclusions added, submission for internal review	INTRA
V0.7	29/04/2021	Internal review	F6S
V1.0	30/04/2021	Addressing internal review comments, updates, finalization for formal submission	INTRA

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	

** 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.

EXECUTIVE SUMMARY

The current document complements the first software release of the COVID-X Sandbox, presenting the services comprising this release. The COVID-X Sandbox introduces a combination of different functions that collectively aim to enable seamless access to a set of healthcare data sources. For this purpose, the Sandbox is capitalising on the ELK Stack components to deliver data management, harmonization/cataloguing, storage, indexing, search, visualization, querying, and retrieval services. Furthermore, it supports the security mechanisms that prevent unauthorised access to personal data based on the well-established encryption protocols and services and on relevant technologies. It is based on a highly modular architecture and each component within the architecture realizes and delivers one or more of the aggregated services. Initially, the document describes the Sandbox CI/CD Stack that will be used for testing and integrating the internal Sandbox services, as well as external applications and components provided by third parties. Subsequently, the services that were developed and the software tools that were employed for realizing the Sandbox services of the 1st release are presented. Finally, the document provides an overview of the available Sandbox deployments and a user guide for administrators on installing and configuring the Sandbox internal services.

TABLE OF CONTENTS

Contents

LIST OF FIGURES

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

LIST OF TABLES

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

ABBREVIATIONS

Accelerate	Transfer technical, business, and ethical knowledge
AI	Artificial Intelligence
API	Application Programmable Interface
CD	Continuous Deployment
CI	Continuous Integration
COVID-X	Innovation action supported by the European Commission in the framework of the EC call SC1-PHE-CORONAVIRUS-2020-2B
CSV	Comma-Separated Values
DB	Database
DSL	Domain Specific Language
ELK	Elasticsearch Logstash Kibana
ES	Elasticsearch
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICH	Instituto Clinico Humanitas
JSON	JavaScript Object Notation
KI	Karolinska Institute
MoSCoW	MUST, SHOULD, COULD, WOULD
OC	Open Calls
OWID	Our Word In Data
RBAC	Role-Based Access
REST	Representational State Transfer
SERMAS	Servisio Madrileno de Salud

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

Sigle Solution (SS)	Open call line for technology providers
Single player	*See Single Solution*
Team Solution (TS)	Open call line for clinical partners working in a team with technology providers
TLS	Transport Layer Security
VCS	Version Control System
WHO	World Health Organization
XML	Extensible Markup Language

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

1 Introduction

1.1 Purpose of the Deliverable

The current document complements the first software release of the COVID-X Sandbox, presenting the services comprising this release. The main challenge of the COVID-X Sandbox is to integrate healthcare datasets from various sources. These datasets are characterized by multiple types of heterogeneity. For this purpose, the Sandbox capitalizes on the ELK Stack components to deliver data management, harmonization/cataloguing, storage, indexing, search, visualization, querying, and retrieval services. It is based on a highly modular architecture, as presented in *D2.1 - Sandbox design & Datalake creation and ingestion* [1], and each component within the architecture realizes and delivers one or more of the aggregated services. The Sandbox realizes security mechanisms that prevent unauthorized access to personal data based on the well-established encryption protocols and services, and relevant technologies. Furthermore, the Sandbox implements an API gateway responsible for exposing the Sandbox services to external entities (users or applications) in a controlled and secure way. In addition to that, it supports load balancing services that ensure high availability and increased performance.

The integration of the multiple COVID-X services follows the best practices of the Continuous Integration and Continuous Deployment (CI/CD) process that enables development teams to deliver code changes more frequently and reliably. COVID-X Sandbox supports a centralized cloud-based deployment, capitalizing on virtualization and orchestration technologies. In addition to that, a local deployment on the premises of the data providers/clinical partners can be adopted as a solution in order to tackle the closed anonymized data access/distribution issues and ensure GDPR and national regulations compliance. In this deployment scenario, data providers must provide a local infrastructure that will be able to host a local instance of the Sandbox.

The first release of the Sandbox services is published on the COVID-X Gitlab repository and will be used for integrating the software applications and services of third parties onboarded on the first COVID-X Open Call.

1.2 Structure of the Deliverable

The deliverable is structured as follows:

- **Chapter 1** introduces the deliverable and explains its overall purpose and structure.
- **Chapter 2** describes the Sandbox CI/CD Stack that will be used for testing and integrating the internal Sandbox services, as well as external applications and components provided by third parties.
- **Chapter 3** presents the developed services and the software tools that were employed for realizing the Sandbox services.

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

- **Chapter 4** provides an overview of the centralized cloud-based deployment of the Sandbox. Moreover, it introduces the available infrastructure at the COVID-X clinical partners' premises that are aimed to be used for hosting the Sandbox local deployments.
- **Chapter 5** provides a user guide to administrators for installing and configuring the Sandbox services.
- **Chapter 6** concludes the document and defines follow-up activities for the final release of the Sandbox, as well as for the first integration of the third parties.

2 Sandbox CI/CD Stack and Integration Approach

The COVID-X Sandbox is implemented as a combination of different functions that aim to enable seamless access to a set of healthcare data sources and offer core data management, security, visualization and federated learning services to third parties to use while integrating their solutions. Various services must be integrated and work collectively for this purpose, forming a layered and modular architecture, as described in *D2.1 – Sandbox design & Datalake creation and ingestion* [1]. The COVID-X Sandbox offers a Continuous Integration (CI) and Continuous Deployment (CD) stack that enables the integration of different services into the Sandbox and performs system testing activities. The CI process enables the extensive testing of each software component/module, as well as the integrated Sandbox platform as a whole. Automated, per component tests and combined integration tests performed on each new version of a component shall be executed, and upon success, the CD process will update the integrated platform by means of deployment of the new versions of the modules. This chapter presents the software stack that supports the CI/CD workflows. The CI/CD Stack has been successfully deployed by month 2 of the project.

2.1 Introduction to the Sandbox CI/CD Stack

The Sandbox CI/CD stack is comprised of several open-source software tools, which collectively aim to create an automated build system capable of integrating changes performed by developers working on different services of the Sandbox. In addition to that, the Sandbox CI/CD stack acts as a testing and system integration environment for effectively testing, integrating, and deploying the software services of the Sandbox in a stable operational environment.

The key features of the Sandbox CI/CD stack are the following:

- The tools that comprise the Sandbox CI/CD stack are running in Docker containers, allowing the whole CI/CD stack to be flexible and independent of the underlying physical infrastructure.
- Deploying the software components and services in a distributed way using Docker makes it easier to scale the stack by introducing additional resources where necessary horizontally.
- Secure communication among the deployed software components achieved by encrypted communications over TLS/HTTPS protocols.
- Isolation and protection from unauthorized access hosts thanks to established authentication and authorization policies and rules.
- Triggerable events (commit, merge, etc.) that automatically initiate building, test, and deployment pipelines with additional support for multiple environments (development, test, production) and with multiple branches per each.

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

- Protected branches policy that ensures that collaborators on a repository cannot make irreversible changes to branches.

The software components that synthesize the Sandbox CI/CD platform are the following:

- **Gitlab**[1]: An easy yet powerful and intuitive git Version Control System (VCS). It offers a web-based graphical interface with several built-in features, such as version control, issue tracking, code review, wiki, etc. Multiple developers can concurrently create, merge and delete parts of the code they are working on independently at their local system before applying the changes to the shared GitLab repository.
- **Jenkins**[2]: An open-source automation server that acts as the CI/CD server, responsible for automating the parts of software development related to building, testing, and deployment of applications. Some of the tasks that can be automated through Jenkins include software builds, unit testing, packaging, and pushing of container images to the Docker Registry.
- **Docker Daemon**[3]: Services running on multiple development virtual servers for the deployment of containerized services. Docker Daemon is a background process that manages Docker images, containers, networks, and storage volumes. The Docker Daemon constantly listens to Docker API requests and processes them.
- **JFrog Docker Registry**[4]: A server-side application that stores and lets you distribute Docker images. It is used to securely control where the container images are being stored and how they are distributed.
- **Portainer**[5]: An open-source tool for managing container-based applications in various virtualization environments. It can be used to set up and manage the environment, deploy applications, monitor application performance, triage problems, and enable role-based access control.

2.2 CI/CD Workflows

The Sandbox CI/CD Stack supports two different kinds of workflows: the development workflow and the deployment workflow. The development workflow is utilized when development teams implement changes on the source code of a Sandbox service. This workflow builds the new version of the code and runs the defined tests on a development and testing environment. On the other hand, deployment workflow is used when a new stable release of a Sandbox service is published. This workflow pulls the latest stable release and deploys the service to the defined operational environment.

[1] https://gitlab.com/
[2] https://www.jenkins.io/
[3] https://www.docker.com/
[4] https://jfrog.com/
[5] https://www.portainer.io/

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

2.2.1 Development workflow

Figure 1 depicts an example CI/CD workflow during the development phase of the Sandbox services. In this scenario, developers (from their premises) can commit source code changes, implementing new component features or integration endpoints, and push their code to GitLab, the central source code repository in this environment. This event automatically triggers Jenkins to pull, compile, build and test the latest version of the source code from the specific feature branch of the code repository. If the unit tests finish successfully, Jenkins will use Docker to create a Docker image, which will then be pushed to the private JFrog Docker Registry with an appropriate tag. Finally, once components have been built and their images have been pushed to the Docker Registry, they are available to be pulled from any server with access to the said registry. This final step can be included in the Jenkins pipeline to deploy the newest version of the application to a predefined development environment, allowing further system integration and User Acceptance (UA) tests to be executed.

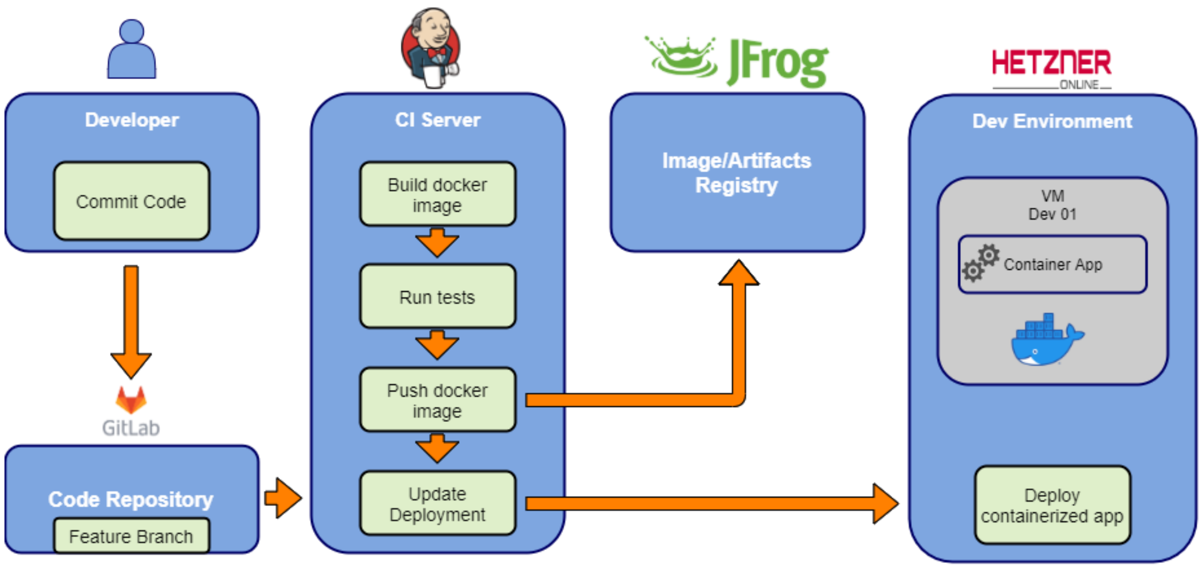


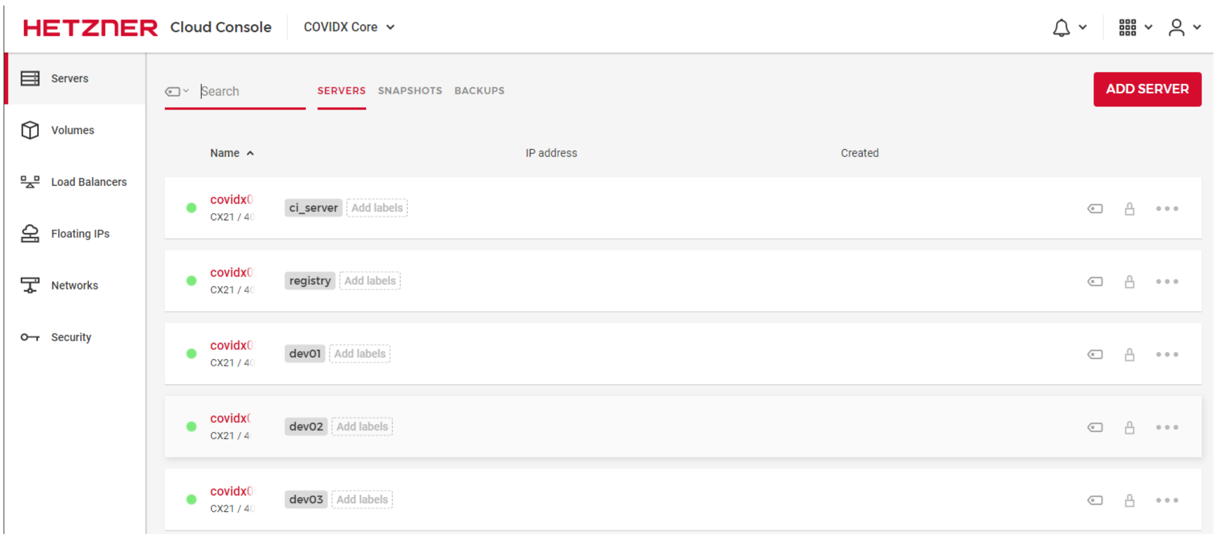
FIGURE 1: CI/CD DEVELOPMENT WORKFLOW

The development environment of the Sandbox is hosted on virtual cloud servers that have been instantiated on a public cloud provider (Hetzner Cloud[6]), as depicted in Figure 2. This environment is mainly used for the development and testing purposes of the Sandbox services. It offers an isolated workspace that shall allow the developers to assess how their features work together and try out improvements. Preliminary testing will happen in this stage. This method prohibits the release of

[6] https://www.hetzner.com/

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

components before the developers make sure that their components work and reduces the number of bugs before they proceed to the demonstration phase.



Name		IP address	Created
covidx CX21 / 4	ci_server Add labels		
covidx CX21 / 4	registry Add labels		
covidx CX21 / 4	dev01 Add labels		
covidx CX21 / 4	dev02 Add labels		
covidx CX21 / 4	dev03 Add labels		

FIGURE 2: SANDBOX DEVELOPMENT ENVIRONMENT

By adopting this approach, developers are able to develop, debug and deploy their software components independently by defining their own automated pipelines. This way, when a software component needs to be updated and redeployed, the pipeline of this component is triggered, and the new updates and changes to the component automatically become available.

2.2.2 Deployment workflow

Figure 3 depicts the deployment workflow that is executed when the development of a new feature is completed and a new version of the software application is ready to be released. In this scenario, a developer creates a new pull request to merge all the changes in the source code introduced from the feature branch to the master branch of the repository. This event automatically triggers Jenkins to execute a new pipeline, downloading the latest docker image from the private docker registry and deploying the new version of the application on the defined operational environment.

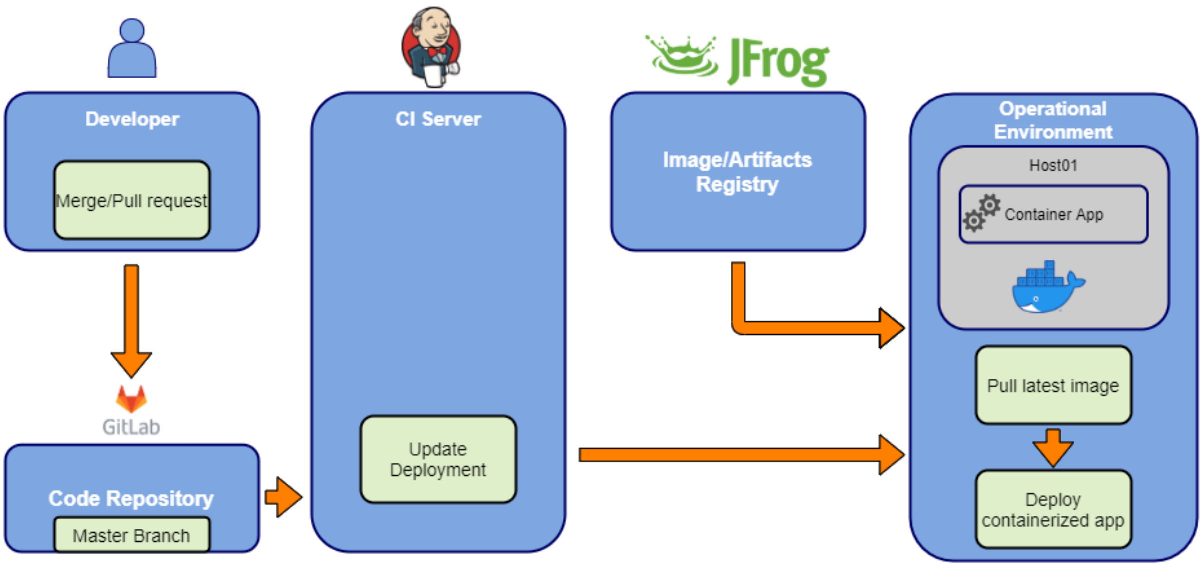


FIGURE 3: CI/CD DEPLOYMENT WORKFLOW

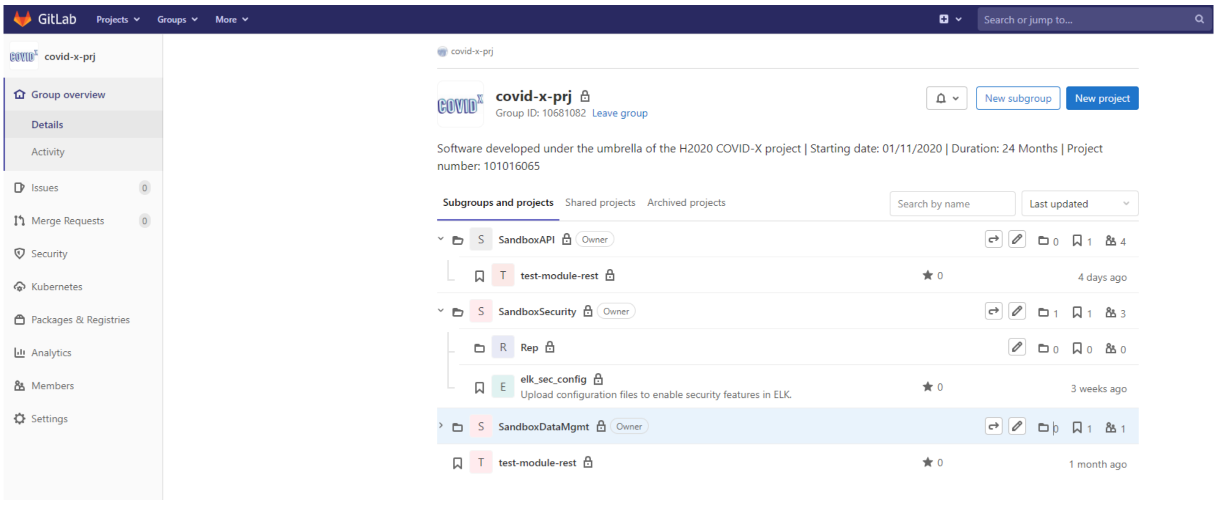
2.3 Version Control System

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems (VCS) are tools that host remote repositories of files that comprise the source code of a software application. This approach helps development teams to keep track of every modification to the source code. Gitlab is used as VCS in the COVID-X Sandbox CI/CD stack. Multiple developers can concurrently create, merge and delete parts of the code they are working independently at their local system before applying the changes to the shared GitLab repository.

A dedicated Gitlab group named "COVID-X-PRJ"[7] has been created, as depicted in *FIGURE 4*. Under this group, several sub-groups and repositories that will host the source code of the Sandbox services are created. Each partner has been granted access to the appropriate sub-groups in order to create repositories, organize users in teams and upload their source code.

[7] https://gitlab.com/covid-x-prj

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



covid-x-prj

Group ID: 10681082 Leave group

Software developed under the umbrella of the H2020 COVID-X project | Starting date: 01/11/2020 | Duration: 24 Months | Project number: 101016065

SandboxAPI Owner 0 1 4
test-module-rest 0 4 days ago
SandboxSecurity Owner 1 1 3
Rep 0 0 0
elk_sec_config 0 3 weeks ago
Upload configuration files to enable security features in ELK.
SandboxDataMgmt Owner 0 1 1
test-module-rest 0 1 month ago

FIGURE 4: COVID-X GITLAB PROJECT

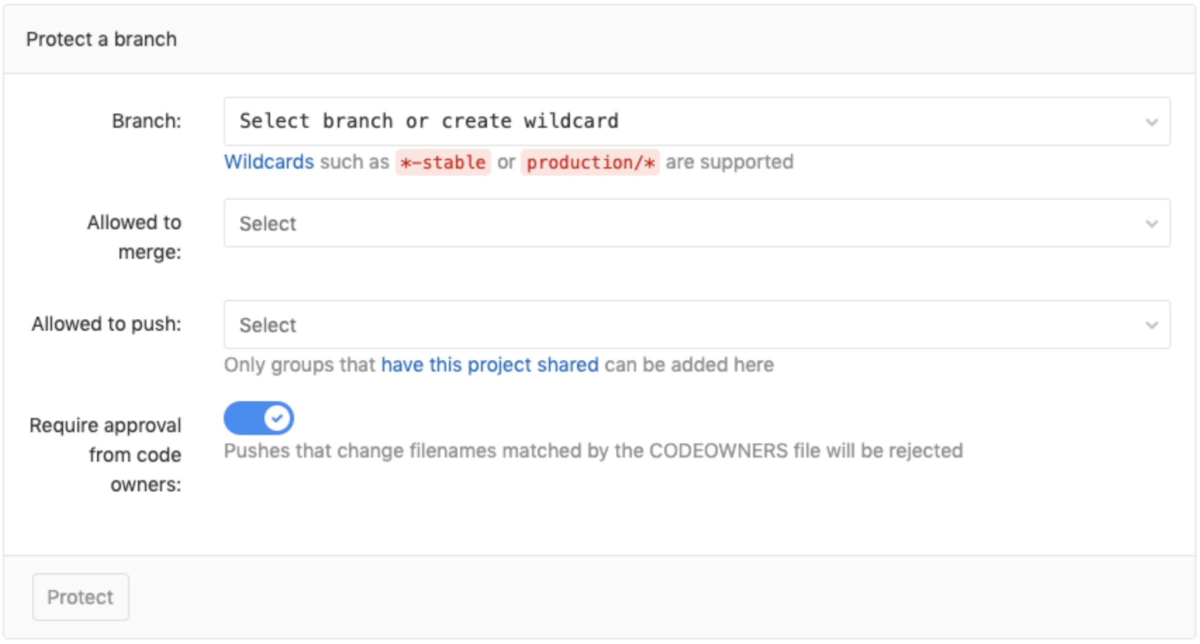
Each Gitlab repository should include the following configuration files in order to support the deployment of dockerized application through the CI/CD workflows:

- **Jenkinsfile**: A text file that contains the definition of a Jenkins Pipeline, including the steps that must be followed during the CI/CD process.
- **Dockerfile**: A text-based script of instructions that is used to create a container image.
- **docker-compose.yml**: A YAML configuration file for defining and running multi-container Docker applications.

Like every other git VCS, Gitlab comes with extended branching capabilities. In most cases, there is one main branch in a repository from which each developer who works on a specific feature or bug fix creates an additional, diverging branch. Once the developers have concluded their changes on the source code, they merge their side branch back into the main branch.

By default, any pull request can be merged at any time. This approach, however, introduces a risk: Developers may inadvertently push insecure content onto the main branch. By enabling protected branches, you can choose to enforce restrictions on how a pull request is merged into a repository, ensuring that collaborators cannot make irreversible changes to branches. In addition to that, it can enable other optional checks and requirements, such as requiring pull request reviews or certain status checks to pass before allowing a pull request to merge. Protected branches policy is enabled by creating branch rules in a repository for a specific branch, all branches, or any branch that matches a naming pattern, as depicted in *FIGURE 5*.

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



Protect a branch

Branch: Select branch or create wildcard

Wildcards such as `*-stable` or `production/*` are supported

Allowed to merge: Select

Allowed to push: Select

Only groups that have this project shared can be added here

Require approval from code owners: ✓

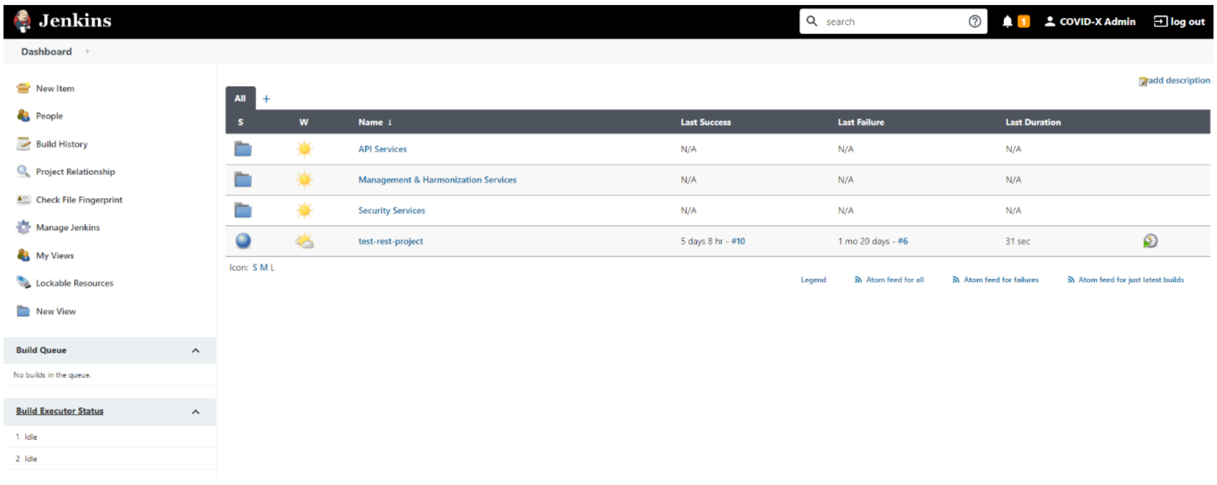
Pushes that change filenames matched by the CODEOWNERS file will be rejected

FIGURE 5: GITLAB PROTECTED BRANCHES

2.4 Automated build and testing

Continuous integration (CI) is a software development practice that enables frequently committing code to a shared repository. When developers, as team members, commit their code changes into a central repository, they must ensure that these changes do not introduce any errors. CI process automates the integration of code changes from multiple contributors into a single software project. A shorter feedback loop means having more iterations, where each iteration introduces automated builds and unit tests on the latest code changes to immediately detect any errors, improving the overall software quality.

Building and testing source code projects in an automated manner requires a CI server. In the Sandbox CI/CD stack, Jenkins is used as the system CI server, running inside a Docker container. Every repository existing under the COVID-X group in Gitlab will be connected to a Jenkins Pipeline, like the ones depicted in *FIGURE 6*.



Jenkins

Dashboard

- New Item
- People
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views
- Lockable Resources
- New View

Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

S	W	Name ↓	Last Success	Last Failure	Last Duration
		API Services	N/A	N/A	N/A
		Management & Harmonization Services	N/A	N/A	N/A
		Security Services	N/A	N/A	N/A
		test-rest-project	5 days 8 hr - #10	1 mo 20 days - #6	31 sec

Icon: S M L

FIGURE 6: SANDBOX JENKINS DASHBOARD

A CI pipeline includes the following steps:

i. Developers implement changes to the source code on their local systems.

ii. They commit the changes and push the new source code to the shared repository.

iii. CI server receives a notification from the VSC system when a push event occurs.

iv. CI server pulls the latest version of the source code, builds the software application, and runs the defined unit and integration tests.

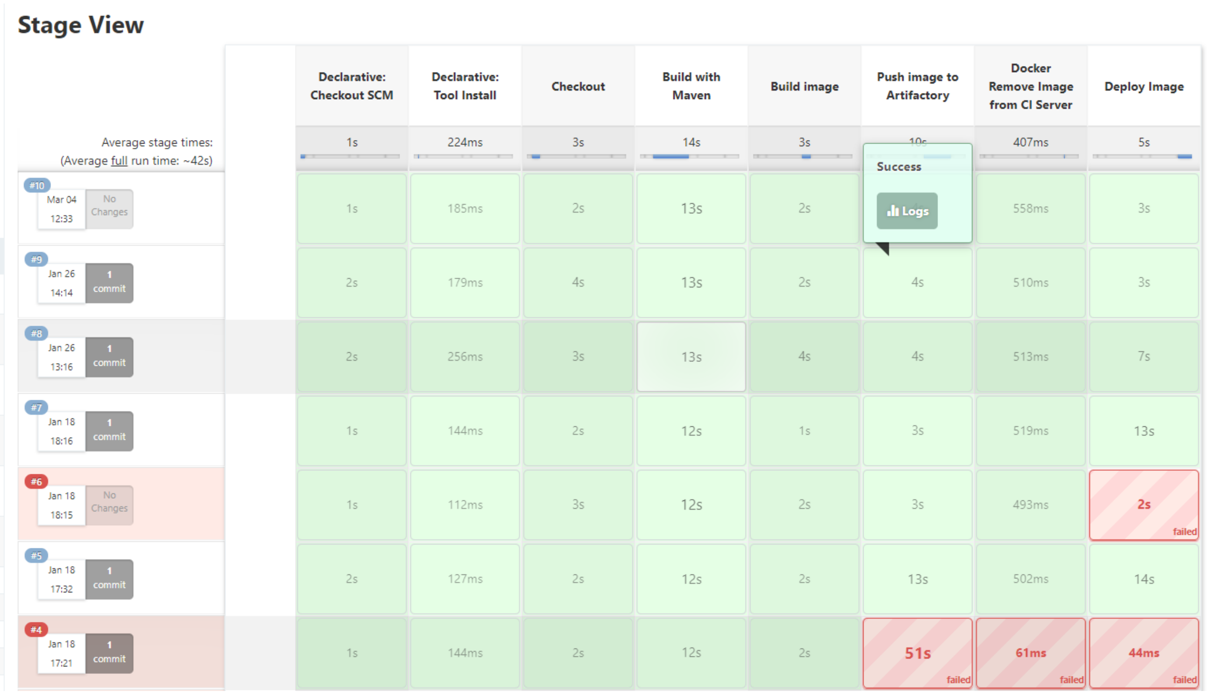
v. If the tests are completed successfully, the CI server assigns a build tag to the code version it just built and pushed the produced artifacts to a registry.

vi. The CI server sends reports of the successful build and tests or alerts in case of build or test failure to the development team.

vii. The team fixes the issues as soon as is possible and runs a new iteration.

A pipeline that includes compilation, build and test stages is shown in *FIGURE 7*. Each pipeline is described in a specific file called Jenkinsfile, which is usually included in the Version Control System repository and is pulled by Jenkins whenever a new build is triggered. Jenkinsfile allows developers to implement pipelines in code formats.

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



Stage View

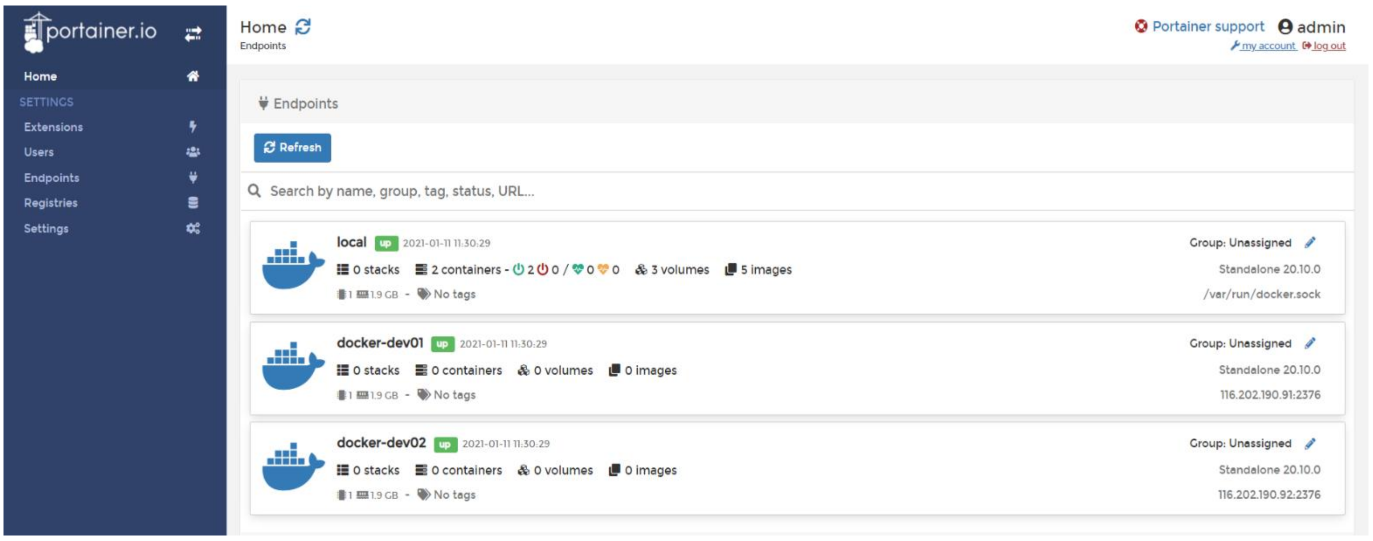
	Declarative: Checkout SCM	Declarative: Tool Install	Checkout	Build with Maven	Build image	Push image to Artifactory	Docker Remove Image from CI Server	Deploy Image
Average stage times: (Average full run time: ~42s)	1s	224ms	3s	14s	3s	10s	407ms	5s
#10 Mar 04 12:33 No Changes	1s	185ms	2s	13s	2s	Success Logs	558ms	3s
#9 Jan 26 14:14 1 commit	2s	179ms	4s	13s	2s	4s	510ms	3s
#8 Jan 26 13:16 1 commit	2s	256ms	3s	13s	4s	4s	513ms	7s
#7 Jan 18 18:16 1 commit	1s	144ms	2s	12s	1s	3s	519ms	13s
#6 Jan 18 18:15 No Changes	1s	112ms	3s	12s	2s	3s	493ms	2s failed
#5 Jan 18 17:32 1 commit	2s	127ms	2s	12s	2s	13s	502ms	14s
#4 Jan 18 17:21 1 commit	1s	144ms	2s	12s	2s	51s failed	61ms failed	44ms failed

FIGURE 7: JENKINS PIPELINE STAGES

2.5 Portainer

Portainer is an open-source software component that runs in a single Docker container. It offers a lightweight web User Interface (UI) that allows the users 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.

In the CI/CD Stack, Portainer is deployed in a Docker container, along with Jenkins CI Server to enable automated deployments on remote Docker environments. *FIGURE 8* depicts the home page of the Portainer that is running on the centralized deployment of the Sandbox on the Hetzner Cloud. The figure presents two Docker hosts of the development environment that have been added to the Portainer panel.



local up 2021-01-11 11:30:29 — 0 stacks, 2 containers - 2 0 / 0 0, 3 volumes, 5 images — 1 1.9 GB - No tags — Group: Unassigned — Standalone 20.10.0 — /var/run/docker.sock

docker-dev01 up 2021-01-11 11:30:29 — 0 stacks, 0 containers, 0 volumes, 0 images — 1 1.9 GB - No tags — Group: Unassigned — Standalone 20.10.0 — 116.202.190.91:2376

docker-dev02 up 2021-01-11 11:30:29 — 0 stacks, 0 containers, 0 volumes, 0 images — 1 1.9 GB - No tags — Group: Unassigned — Standalone 20.10.0 — 116.202.190.92:2376

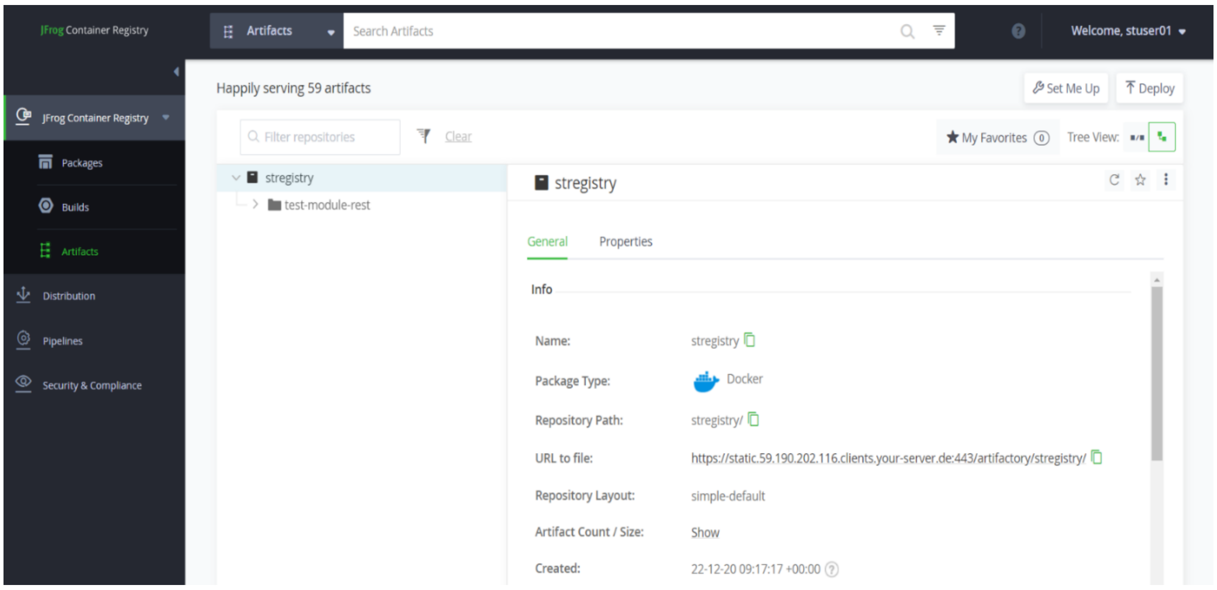
FIGURE 8: PORTAINER HOME SCREEN

2.6 JFrog Docker Registry

Docker registry is a centralized storage and content delivery system that allows users to store, manage, and distribute named Docker images, available in different tagged versions. Users and developers interact with the registry by using the Docker push command to store a new image and pull command to download a stored image. Private Docker registries allow Docker images to be stored and distributed in a controlled and safe way.

In the COVID-X Sandbox, the CI/CD stack JFrog serves as the private Docker registry. Developers and users of the Sandbox services can use this registry in order to push/pull their Docker images. JFrog is deployed in a Docker container and offers a web UI for monitoring and managing the registry and the stored Docker images. *FIGURE 9* presents the web UI of the deployed JFrog registry on the Hetzner Cloud.

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



Name: stregistry
Package Type: Docker
Repository Path: stregistry/
URL to file: https://static.59.190.202.116.clients.your-server.de:443/artifactory/stregistry/
Repository Layout: simple-default
Artifact Count / Size: Show
Created: 22-12-20 09:17:17 +00:00

FIGURE 9: COVID-X JFROG DOCKER REGISTRY

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

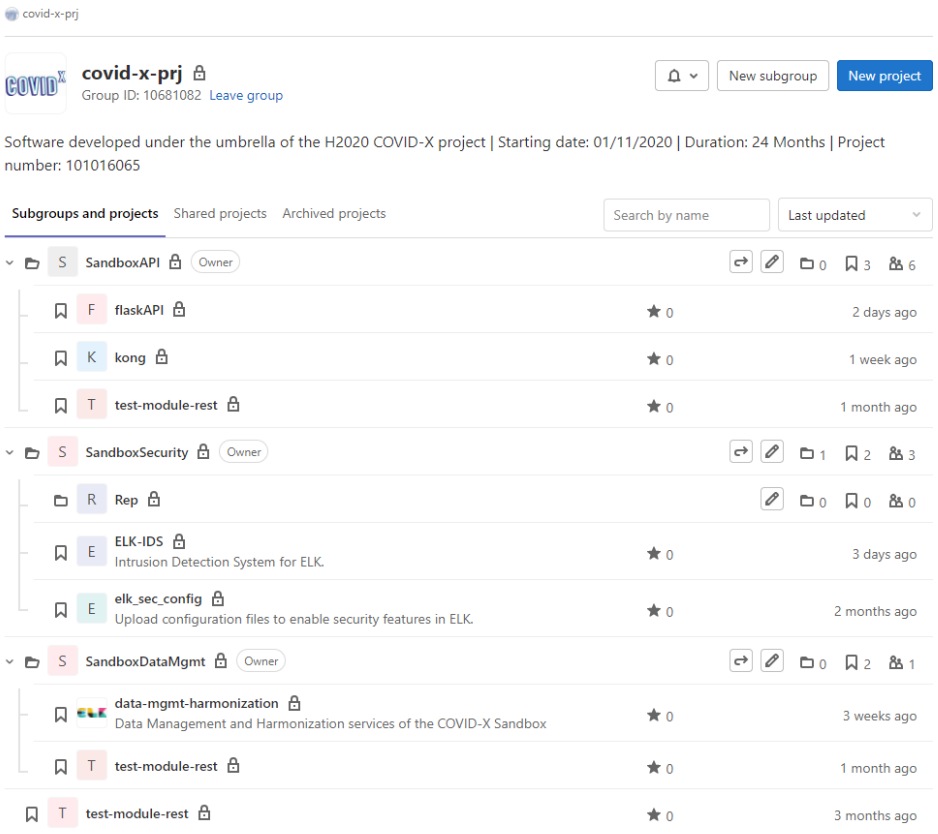
3 Sandbox Integrated Services – Release A

This chapter describes the implementation approach that was followed for developing, testing, and integrating services comprising the first release of the Sandbox. The first section provides an overview of the Sandbox architecture, while the subsequent sections describe the technologies and software applications that have been employed for the development of the Sandbox services.

3.1 Overview of the Sandbox Architecture – Release A

The Sandbox is based on a highly modular architecture, built as a collection of microservices, following the Service-Oriented Design approach, as extensively described in *D2.1– Sandbox design & Datalake creation and ingestion* [1] . By adopting the SOA approach, the Sandbox functionalities are divided into three specific categories of services that are independently developed by different development teams, following the CI/CD workflows described in the previous chapter. These categories are (i) Data Ingestion, Management, Harmonization and Visualization services, (ii) Security services, and (iii) Sandbox interfaces. A dedicated Gitlab group has been assigned to each development team responsible for the development of the respective Sandbox service, as depicted in Figure 10.

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


covid-x-prj

covid-x-prj
Group ID: 10681082 Leave group

New subgroup New project

Software developed under the umbrella of the H2020 COVID-X project | Starting date: 01/11/2020 | Duration: 24 Months | Project number: 101016065

Subgroups and projects | Shared projects | Archived projects | Search by name | Last updated

S SandboxAPI (Owner)	0 folders, 3 projects, 6 members		
F flaskAPI	★ 0	2 days ago
K kong	★ 0	1 week ago
T test-module-rest	★ 0	1 month ago
S SandboxSecurity (Owner)	1 folder, 2 projects, 3 members		
R Rep	0 folders, 0 projects, 0 members		
E ELK-IDS — Intrusion Detection System for ELK.	★ 0	3 days ago
E elk_sec_config — Upload configuration files to enable security features in ELK.	★ 0	2 months ago
S SandboxDataMgmt (Owner)	0 folders, 2 projects, 1 member		
data-mgmt-harmonization — Data Management and Harmonization services of the COVID-X Sandbox	★ 0	3 weeks ago
T test-module-rest	★ 0	1 month ago
T test-module-rest	★ 0	3 months ago

FIGURE 10: SANDBOX REPOSITORIES ON GITLAB

These services are integrated and combined to form the Sandbox architecture for the first release on month 6 of the project, as presented in Figure 11, and already documented in *D2.1 – Sandbox design & Datalake creation and ingestion* [1]. The communication among the different services is realized through RESTful endpoints exposed internally, based on the technology and programming language that has been employed for the implementation of each service. These endpoints expose information about the service's resources and allow client services to perform specific actions on those resources. These actions are defined with specific HTTP methods, such as GET, POST, PUT and DELETE, while the JSON format is used for exchanging data.

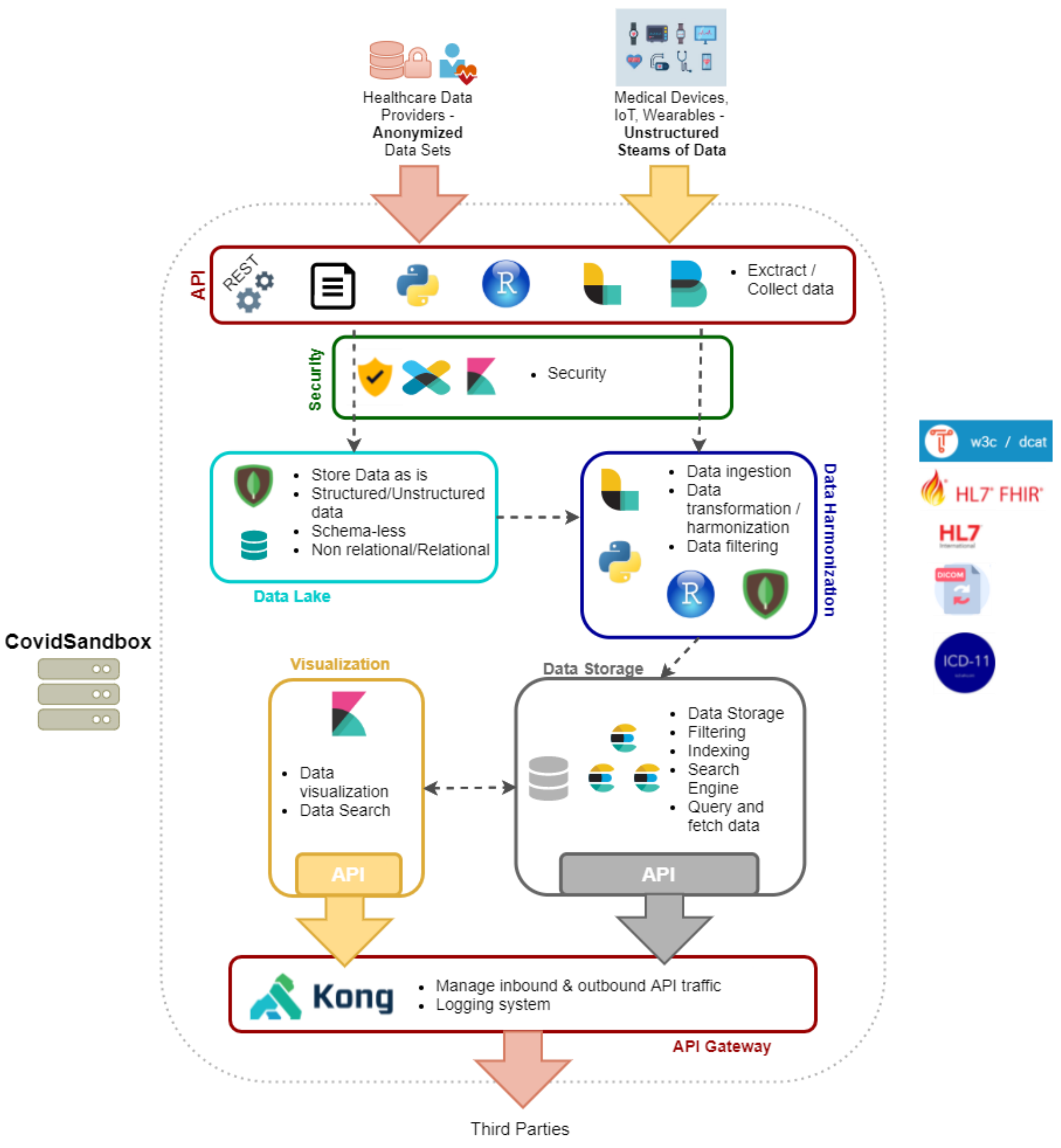


FIGURE 11: SANDBOX RELEASE A ARCHITECTURE

3.2 Data Integration, Management, Harmonization, and Visualization Services

The Sandbox capitalizes on the ELK Stack applications to deliver data management, harmonization, cataloging, storage, indexing, search, visualization, querying, and retrieval services. The Data Management and Harmonization service follows a microservices approach, being further divided into

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

multiple building blocks. Each building block is small in size and autonomously developed, realizing a specific microservice. Similarly to the other services of the Sandbox, these microservices are also deployed in Docker containers. The following subsections present the technologies utilized for realizing the data management and harmonization service.

3.2.1 Elasticsearch

Elasticsearch[8] is based on Apache Lucene[9] open-source search framework. It is always deployed in a distributed manner, forming a cluster that efficiently performs data management tasks, such as full-text search and structure data indexing, aggregation, filtering, and cataloging. In Elasticsearch, data are encoded following a key-value pair format in JSON objects called documents. In terms of relational databases, a document is similar to an SQL table row, representing a specific entity. Each field in the document is conceptually related to an SQL table column. Every document has a unique id that can be either explicitly assigned by the user or generated by Elasticsearch.

Documents are organized in indices. An index is the highest-level entity that users can query in elasticsearch, and it contains a collection of logically related documents. Indices define the scheme of the stored data, as well as configuration options, such as scalability and availability. In terms of relational databases, indices are corresponding to SQL tables. For each available dataset, a specific index or index group (multiple indices that follow a specific naming pattern) is created to host the data collected from the dataset.

One of the key features of Elasticsearch is its extensive REST API which allows users to integrate, manage and query the indexed data in multiple different ways. Moreover, these APIs can be directly called by administrators to configure and access Elasticsearch features. Elasticsearch provides a full Domain Specific Language[10] (DSL) based on JSON to define queries.

The Sandbox deployments include an Elasticsearch cluster to realize part of the data management and storage services, namely storing, indexing, and searching for data. Moreover, Elasticsearch offers dataset cataloging capabilities. A new index or a group of indices following a common pattern is created for each dataset ingested into the Sandbox. The data schema of each index is defined following the specifications of each dataset, while the ingested data are harmonized and annotated based on the COVID-X semantic data model defined in D2.1.

Elastic nodes are instantiated within Docker containers that may run either over the same server, forming an all-in-one deployment, or in a distributed manner over multiple servers. The initial configuration options of the cluster are defined through environmental variables in a "dotenv" configuration file. More elasticsearch nodes can be added to the cluster on run time, depending on the requirements of each use case. The new nodes add storage capacity to the cluster and improve the system's throughput for handling queries.

[8] https://www.elastic.co/elasticsearch/
[9] https://lucene.apache.org/
[10] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

3.2.2 Kibana

Kibana[11] acts as a web user interface for Elasticsearch. It comprises an analytics and visualization platform, which allows users to process, visualize and analyze data stored in Elasticsearch efficiently. It sends queries utilizing the standard Elasticsearch REST API, while it provides an interface for building those queries and configure how to display the results. Kibana includes a set of applications that allows users to create custom visualizations, such as pie charts, line charts, heat maps, and gauges, organized in fully customizable Dashboards. In addition to the querying, visualization, and analytics capabilities, Kibana allows users to configure certain parts of the ELK stack features, including the security mechanisms. These mechanisms allow basic user authentication and authorization, as well as Role-Based Access management, described in more detail in section 3.3.

The Sandbox utilizes Kibana to create a dashboard for the data management services. Through this dashboard, users can visualize and analyze data stored in the Elasticsearch cluster. For this reason, multiple dashboards with the appropriate access permissions are created and assigned to the third-party users. These dashboards are configured to include multiple visualizations according to the needs of each third-party use case. Kibana is deployed in a Docker container, running in the same server that hosts the Elasticsearch master node. It communicates with the other services of the ELK stack over a shared Docker network that enables encrypted traffic with TLS/HTTPS protocol. Kibana exposes a web user interface that is reachable directly by the external users. The web UI is utilized for creating custom visualization and dashboards per use case, like the one depicted in **Error! Reference source not found.** T hird-party users will be granted access to specific dashboards that comply to their access permissions.

[11] https://www.elastic.co/kibana

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

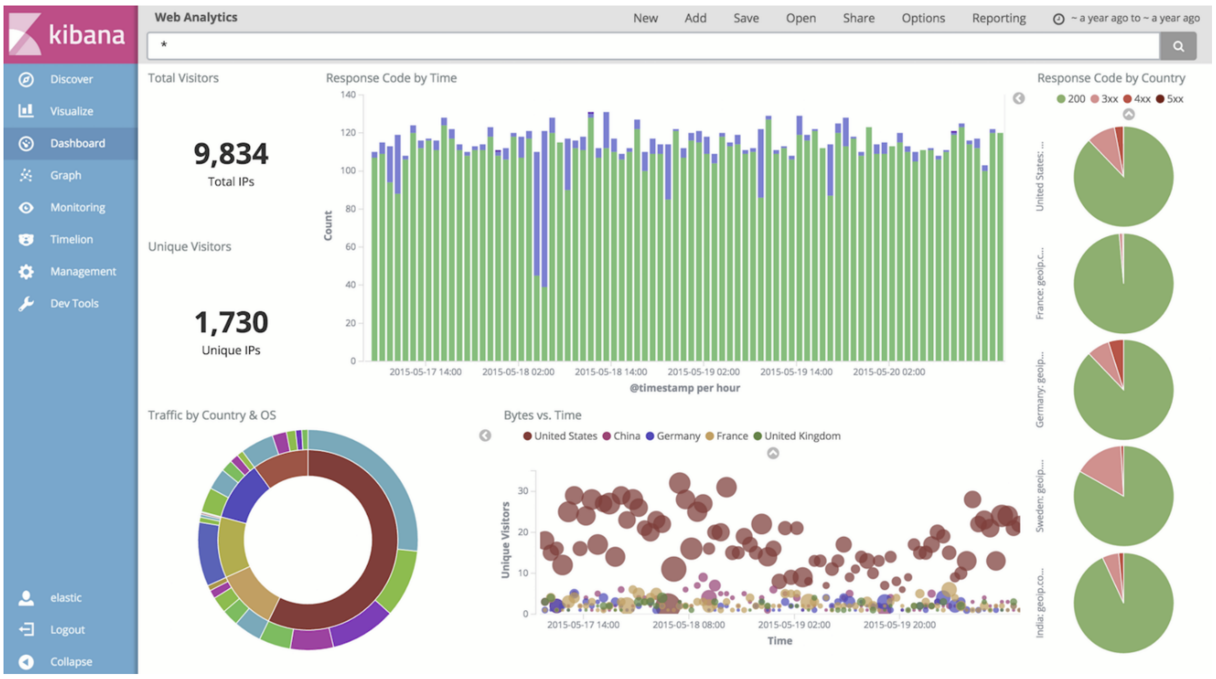


FIGURE 12: KIBANA CUSTOM DASHBOARD

3.2.3 Logstash

Logstash[12] is an open-source data collection and processing engine. It is a plugin-based system that provides a flexible and powerful way to parse and process most types of structured data collected from various heterogeneous data sources. The data that Logstash collects are handled as events that are processed and redirected to one or multiple destinations. Any type of event can be enriched and transformed with a broad array of input, filter, and output plugins, with many native codecs, further simplifying the ingestion process.

A Logstash pipeline consists of three stages, namely input, filter, and output, where each stage is implemented as a collection of plugins. Input plugins read data from the queue in micro-batches and process these through the predefined filter plugins in sequence, transforming them into the required format. Once the data have been processed, the processing threads forward the data to the appropriate output plugins responsible for formatting and sending data to one or more defined destinations, such as Elasticsearch. Logstash pipelines are created based on configuration files, while a Logstash service can simultaneously implement multiple pipelines.

The Sandbox utilizes Logstash pipelines for collecting and integrating data from various heterogeneous data sources. For each data source, a specific pipeline is configured responsible for: (i) extracting the data out of the data source, (ii) processing, annotating, and transforming them in a form that complies with the defined common data model, (iii) sending them to the appropriate index in the Elasticsearch

[12] https://www.elastic.co/guide/en/logstash/current/introduction.html

cluster. Depending on the use case requirements, Logstash can be deployed in one or more Docker containers. These containers communicate with each other and other ELK stack services over a shared Docker network that enables encrypted traffic with TLS/HTTPS protocol.

3.2.4 MongoDB

MongoDB[13] is an open-source document-oriented database that can be used for storing a massive amount of data. It is a NoSQL database, storing data in JSON formatted documents instead of SQL tables and rows. Documents consist of key-value pairs called data fields, which are the basic unit of data in MongoDB. A document in MongoDB corresponds to a row in an SQL database. A group of logically related documents is known as a collection, which is roughly synonymous with an SQL table in terms of a relational database.

MongoDB is a schema-less database, as it does not force any specific data schema on the stored documents. Collections may contain multiple documents with different structures, making the whole system flexible and adaptable to the requirements of each use case. Moreover, documents support nested fields and hierarchical relationships, allowing the efficient representation and storage of complex data structures.

The Sandbox takes advantage of the characteristics described above, utilizing MongoDB to realize the internal data lake. This data lake is used for supporting read/write-heavy applications, with massive workloads that store a vast amount of data. The Sandbox relies on MongoDB services to efficiently store both relational and non-relation data, as well as data from semi-structured or unstructured data sources. Based on the characteristics of the available data sources, as well as the needs of each use case, the Sandbox may store data in MongoDB, in the Elasticsearch cluster, or both databases. The metadata and characteristics of any stored data sets in the data lake are also described and cataloged in Elasticsearch.

3.3 Security Services

The security services configured and integrated into the first release of the COVID-X Sandbox are part of the X-Pack[14] extension. They include user authentication and authorization via a role-based access control system, encrypted inter-node communication in a multi-node cluster, and encrypted client communication.

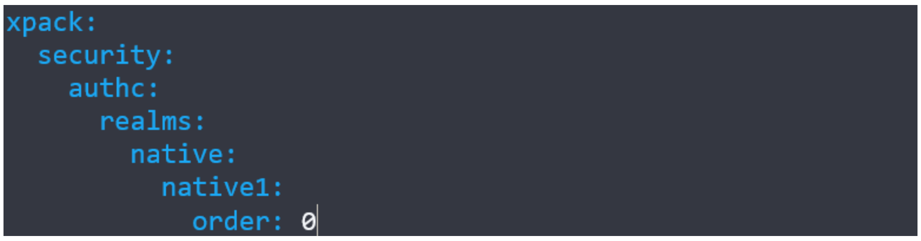
The Elastic Stack authenticates users by identifying the users behind the requests that hit the cluster and verifying that they are whom they claim to be. The authentication process is handled by one or more authentication services called realms. For the first release of the COVID-X Sandbox, the native

[13] https://docs.mongodb.com/manual/introduction/
[14] https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-xpack.html

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

realm is used. The native realm is an internal realm where users are stored in a dedicated Elasticsearch index. This realm supports an authentication token in the form of username and password and is available by default when no realms are explicitly configured.

A realm can be added to the configuration file of elasticsearch (*elasticsearch.yml)* under the *xpack.security.authc.realms.native* namespace. In the case of the 1st release, the native realm is used by default since no other realms are configured. Following this process, it is mandatory for any requests made to the realm to have user credentials attached. An indicative example of configuring a native realm is shown in the figure below:



```
xpack:
  security:
    authc:
      realms:
        native:
          native1:
            order: 0
```

FIGURE 13: CONFIGURATION OF A NATIVE REALM IN XPACK

User management and authentication are feasible with the internal native realm. The Elastic Stack's interfaces provide ways to add and remove users, assign user roles, and manage user passwords. With regards to the Elastic Stack authorization feature, which is the process of determining whether the user behind an incoming request is allowed to execute the request, a role-based access control (RBAC) mechanism is set in place, enabling user authorization by assigning privileges to roles and assigning roles to users. The COVID-X RBAC mechanism is depicted in Figure 14.

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

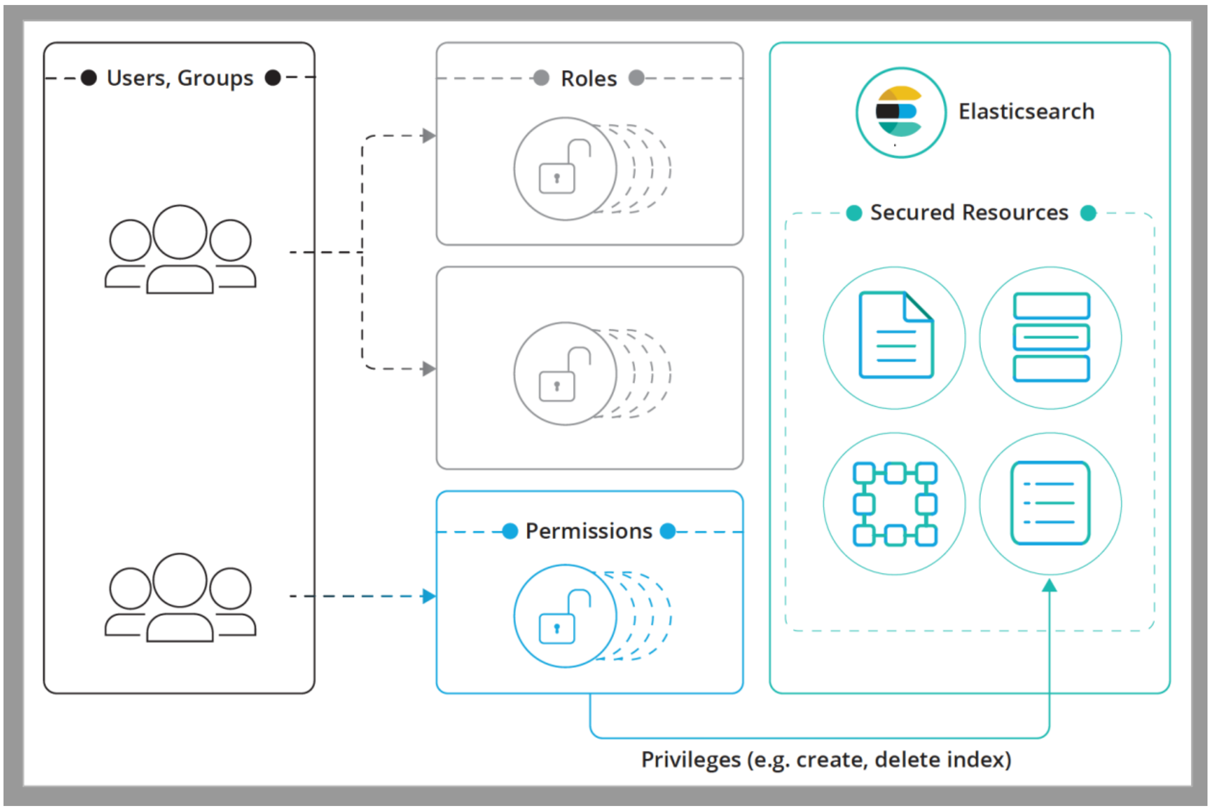


FIGURE 14: COVID-X RBAC MECHANISM

The authorization process revolves around the following constructs:

- **Secured Resource:** A resource to which access is restricted. Indices, aliases, documents, fields, users, and the Elasticsearch cluster itself are all examples of secured objects.
- **Privilege:** A named group of one or more actions that a user may execute against a secured resource. Each secured resource has its own sets of available privileges. For example, *read* is an index privilege representing all actions that enable reading the indexed/stored data. Other Security Privileges available for indices are: *write*, *monitor*, *manage*, *delete*, *create*, etc.
- **Permissions:** A set of one or more privileges against a secured resource. Permissions can easily be described in words; A few examples are listed below:
    - read privilege on the products data stream or index
    - manage privilege on a cluster
    - run_as privilege for a user
    - read privilege on documents that match query X
    - read privilege for a field
- **Role**: A role has a unique name and identifies a set of permissions that translate to privileges on resources. You can associate a user with an arbitrary number of roles. The complete set of permissions that a specific user has is defined by aggregating the permissions in all its roles.

Elastic Stack security features realized through X-Pack enable the encryption of all traffic between HTTP clients and the Elasticsearch cluster, as well as the encryption of all communications between nodes of a cluster, through the establishment of secure connections based on Transport Layer Security (TLS/SSL). TLS requires X.509 certificates to perform encryption and authentication of the application that is being communicated with. In order for the communication between nodes to be truly secure, the certificates must be validated. The certificate authority validation in an Elasticsearch cluster is done by the certificate authority (CA) that signs the certificate. Following this approach, as nodes are added to the cluster, they need to use a certificate signed by the same CA, and the node is automatically allowed to join the cluster.

3.4 Sandbox Interfaces for the Integration of Third Parties

The Sandbox interfaces provide the usual communication capabilities between the client and the system. Most of the ingested dataflows and requests are collected by an API (Application Programming Interface) gateway, which is responsible for redirecting the requests internally to the appropriate Sandbox services. The software selected for this task is the Kong API gateway[15], which is based on NGINX. Kong manages incoming and upstream network traffic efficiently, allowing redirection of the incoming requests. In addition to that, it can act as a message broker. The COVID-X Sandbox exposes a single port to handle the requests by exploring the path included in the URL. Accordingly, these endpoints are provided based on the internal paths and the corresponding security policies. Otherwise, traffic is rejected.

The API services represent all the CRUD operations that may occur in any database. All these operations are either authorized or rejected by the security mechanisms provided by the X-Pack security features of the ELK stack. Specifically, the end-users are allowed to perform data ingestion both in a batch and streaming way, reading, updating, and deleting operations using DSL (ElasticSearch Query: Domain Specific Language). Moreover, every request must have the authorization header provided by the role each end-user has. The following sections detail all this information.

3.4.1 Data ingestion

As it has just been aforementioned, the data ingestion process can follow two different pipelines:

1. **Batch pipeline**: Kong redirects the request to a flask application[16]. Flask is a library in python which offers a simple way to create APIs. The endpoints can be customized by the usage of

[15] Kong API Gateway: https://konghq.com/

[16] Flask Library: https://flask.palletsprojects.com/en/1.1.x/

routes. Like any other service inserted in the Sandbox, this application is dockerized. The whole process followed by a specific request is described as follows:

   a. A POST request is sent to the specific endpoint exposed with Kong; it should contain not only the kong endpoint but also the name of the organization and the index/dataset to which data should be added. Note that the index must exist in advance. This request should have the same credentials as those used in Elasticsearch. The type of request has to be multipart with a file attached and must have the key "file".

   b. The application checks the structure of the request searching for the file included in its body and verifying its extension.

   c. The application sends a request to the elastic cluster using the credentials that have been provided. This request does not query any kind of data; it just gets some metadata of the target index to check if the user has permission to ingest files.

   d. If the response is a 200 status code, the application writes the file into the disk using a hierarchy established as /<org>/<index>/<file>.

2. **Stream pipeline**: the request is directly redirected to the logstash listening port with several requirements:

   a. Each index must have a specific logstash service associated. This way, the authentication can take place.

   b. Kong must have different routes for each of these logstash endpoints, so every port at which the ingestion may occur is mapped to a kong path.

3.4.2 Data update, retrieve and delete

For data update, retrieve, search, and delete operations, Kong directly sends the request to the Elasticsearch cluster, utilizing the available set of APIs. These queries follow the typical DSL[17] syntax and can be either rejected or accepted by the X-pack security extension of the ELK stack. For each operation, the respective API endpoing on Kong is defined.

3.4.3 Kong API Gateway

Kong offers several ways to achieve the desired configuration: (i) specifying a declarative file read at the service instantiation, (iI) using the appropriate REST requests, or (iii) using a GUI (Konga). For the

[17] https://elasticsearch-dsl.readthedocs.io/en/latest/

Sandbox implementation, the declarative file option is promoted. This file is saved in YAML format and allows the configuration of several essential properties, like the following:

- Ensure that Kong does not reach an inconsistent state. The declarative file is either entirely read or entirely ignored in case that an error occurs. This approach is similar to a transaction in an SQL database.
- There is no need to provide an external database for storing all the Kong configuration files. Instead, it is stored in memory, which allows faster responses.

Both of these settings make the API gateway more robust. They enable Kong to be reliably instantiated in a Docker container, preventing requests that might fail to reach the appropriate endpoint. The following figures present the Kong API gateway structure and the web UI enabled by Kong for managing and monitoring the internal services:

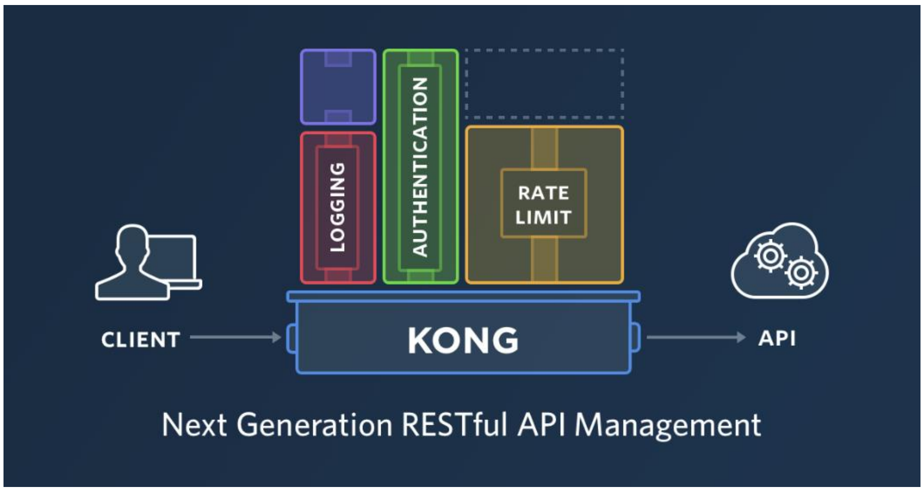
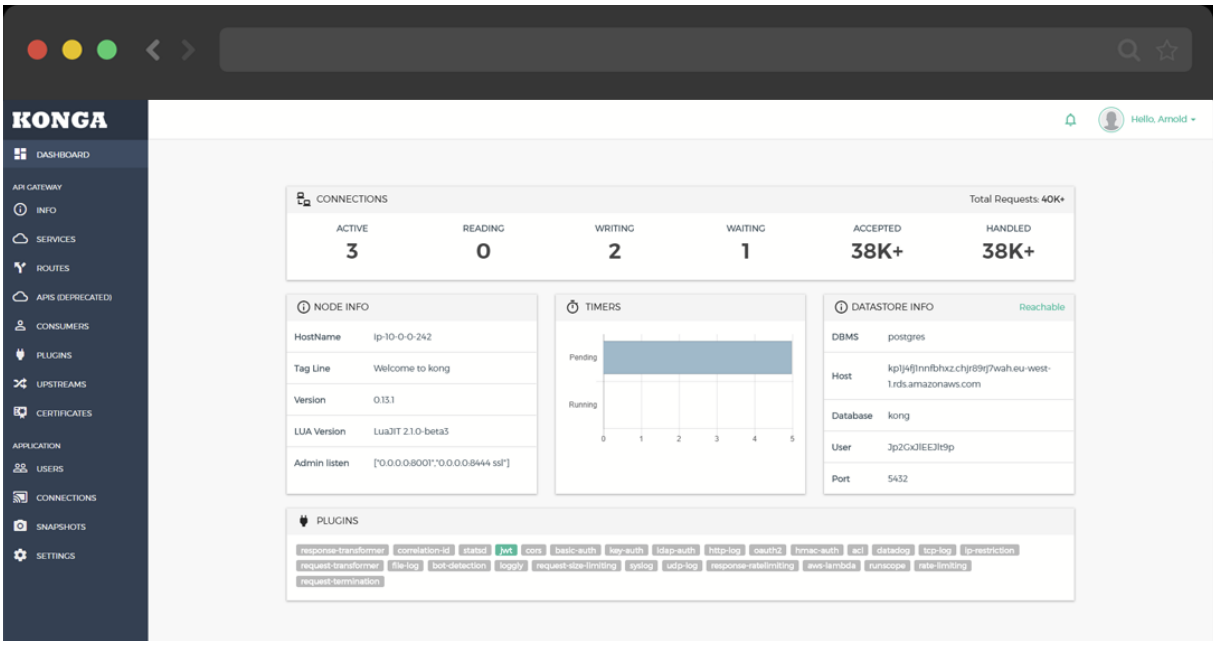


FIGURE 15: KONG API GATEWAY STRUCTURE



CONNECTIONS — Total Requests: **40K+**

ACTIVE	READING	WRITING	WAITING	ACCEPTED	HANDLED
3	0	2	1	38K+	38K+

NODE INFO

HostName	ip-10-0-0-242
Tag Line	Welcome to kong
Version	0.13.1
LUA Version	LuaJIT 2.1.0-beta3
Admin listen	["0.0.0.0:8001","0.0.0.0:8444 ssl"]

DATASTORE INFO — Reachable

DBMS	postgres
Host	kp1j4fj1nnfbhxz.chjr89rj7wah.eu-west-1.rds.amazonaws.com
Database	kong
User	Jp2GxJlEEJlt9p
Port	5432

PLUGINS

response-transformer correlation-id statsd jwt cors basic-auth key-auth ldap-auth http-log oauth2 hmac-auth acl datadog tcp-log ip-restriction request-transformer file-log bot-detection loggly request-size-limiting syslog udp-log response-ratelimiting aws-lambda runscope rate-limiting request-termination

FIGURE 16: KONG GUI FOR SERVICE MANAGEMENT

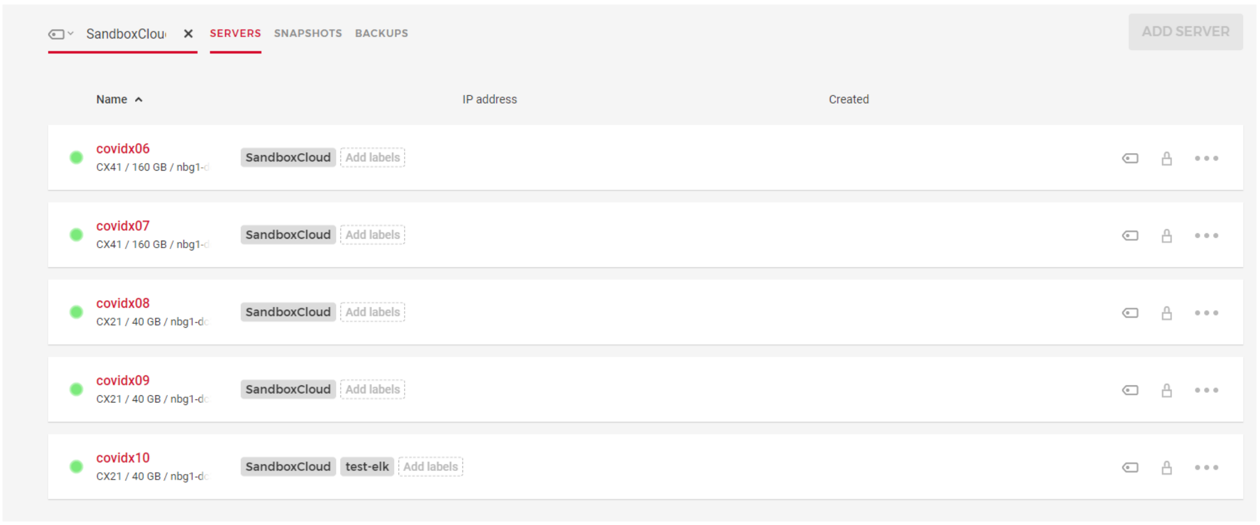
In addition to the API gateway services, Kong provides a load balancer that can be used to redirect the requests to all the nodes established in the cluster. This property is beneficial to the Sandbox, as it allows high availability and increased performance when handling incoming requests. Like the previously mentioned settings, this feature is enabled in the *YAML* declarative configuration file, so all of the benefits of using this approach are preserved. In order for Kong to provide the load balancing service, the endpoints of the nodes that realize the same service must be configured in advance. Load balancing service can be used for adding different traffic weights on each endpoint, based on the specified use case scenario.

4 Sandbox Deployment

This chapter provides an overview of the different Sandbox deployments that are utilized to fulfill both COVID-X Clinical partners needs and legal/operational constraints, as detailed in D2.1, as well as foreseen third-parties' requirements. The first part of this chapter describes the centralized Sandbox deployment in a cloud-based deployment. Subsequent sections describe the process that must be followed when deploying the Sandbox on the clinical partner's premises and provides an overview of the available infrastructures reserved for deploying local Sandbox instances.

4.1 Cloud-based deployment

The centralized instance of the COVID-X Sandbox is deployed on five virtual servers instantiated on Hetzner public cloud, as depicted in Figure 17. It realizes all the core services described in the previous chapter to collect and integrate data from various data sources and provide a visualization panel through a web user interface. Data from the clinical partner's sources that are stored in the cloud-based deployment are fully anonymized, ensuring compliance to GDPR and specific national regulations. 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 these data, selected publicly available datasets are included in the cloud version of the COVID-X Sandbox.



Name		IP address	Created
covidx06 CX41 / 160 GB / nbg1-	SandboxCloud		
covidx07 CX41 / 160 GB / nbg1-	SandboxCloud		
covidx08 CX21 / 40 GB / nbg1-d	SandboxCloud		
covidx09 CX21 / 40 GB / nbg1-d	SandboxCloud		
covidx10 CX21 / 40 GB / nbg1-d	SandboxCloud, test-elk		

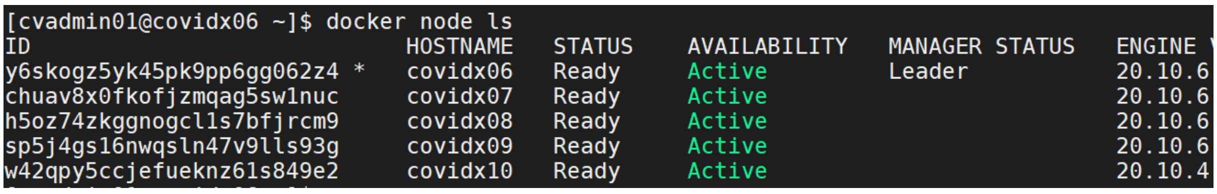
FIGURE 17: HETZNER SERVERS HOSTING SANDBOX CLOUD-BASED DEPLOYMENT

The virtual servers are attached to a Hetzner private network that realizes secure internal communication. In addition to that, each server is assigned a public IP address, enabling external communication. The access is secured and protected by dedicated firewalls running on each server, as well as security policies enabled by the cloud provider, permitting access only to authorized users. Finally, as an extra security measure, servers ensure storage encryption at rest. Table 1 summarizes the provisioned computational resources for the cloud-based deployment of the Sandbox. Additional resources can be appended to the existing ones if required by a specific use case scenario. This can be achieved either by rescaling the already instantiated virtual servers increasing computational power or by adding supplementary servers for accommodating the Sandbox services.

TABLE 1: RESOURCES OF THE SANDBOX CLOUD-BASED DEPLOYMENT

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

The Sandbox services are deployed in a distributed manner running in Docker containers. For this purpose, Docker daemon has been installed on all the virtual servers, configured to run in swarm cluster mode[18], as shown in *Figure 18*. This mode enables multi-host Docker networking, which is necessary in order for the containers running on different servers to be able to communicate with each other directly. 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 18: DOCKER SWARM CLUSTER

[18] https://docs.docker.com/engine/swarm/

Administrator users from the teams responsible for developing the core Sandbox services are responsible for installing and maintaining the cloud-based deployment. The deployment in this operational environment follows the deployment workflow of the CI/CD process described in Section 2.2.2. 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[19] files and wrapper shell scripts. The Table below presents the deployed containers comprising the Sandbox services.

TABLE 2: SANDBOX CONTAINERS

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
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

[19] https://docs.docker.com/compose/

4.2 Local Deployments in Clinical Partners Premises

In addition to the centralized cloud-based deployment of the Sandbox, COVID-X realizes local deployments of the Sandbox running on the premises of the data providers/clinical partners. This solution could be adopted to tackle the closed anonymized data access/distribution issues and ensure GDPR and national regulations compliance, in addition to addressing needs of legal certification of the relevant infrastructure and remain in control of the Data Controllers of the closed health data sets. In this deployment scenario, data providers must provide a local infrastructure that will be able to host a local instance of the Sandbox. Typically for the initial local deployment at the clinical partner infrastructure, a multi-core machine with 64GB of RAM should be available with SSD Drive(s). The disk space has to be a multiple of the volume of the data that will be stored. Multiple hosts would be ideal as a setup of 3x64GB RAM or 3x32GB RAM, or 3x16GB RAM machines would be significantly better and allow for a Highly Available setup. Additional hosts should/could be added depending on the needs of each use case scenario. Clinical partners must allow remote access (SSH) to the machines that will accommodate the necessary Sandbox services, most preferably over a secured VPN tunnel. Administrator privileges will also be required. The general requirements are defined in more details in D2.1.

4.2.1 ICH

Humanitas is providing a Virtual machine that is 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 in order to be able to configure the prerequisites of the COVID-X Sandbox as well as adjust its deployment to the underlying infrastructure. The available hardware infrastructure includes a multi-core machine with 64GiB of RAM and 100GiBs of memory.

4.2.2 SERMAS

Sermas aims to provide a local infrastructure that can host the local deployment of the COVID-X Sandbox, thus ensuring GDPR and national regulations compliance. The initial data samples provided from the clinical partner represent the expected datasets that will be ingested. Sermas has provided ssh access to their system only to certified users through an IPSEC VPN tunnel. A system user has the necessary permissions in order to be able to install and configure the prerequisites of the COVID-X Sandbox as well as adjust its deployment to the underlying infrastructure. The available hardware infrastructure includes a multi-core machine with 128GiB of RAM and 4TBs of memory, allowing for an ideal and highly available setup.

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

4.2.3 KI

Karolinska Institutet provides to the COVIDX project a virtual (VLAN) environment which is hosted in KI's LIME department. Within this environment the test version of the Clinical History Taking program (CLEOS) is operating in a Windows Server. The hardware specifications of the physical server are listed as follows:

- PowerEdge R740/R740XD - Motherboard
- Intel Xeon Gold 6208U 2.9G, 16C/32T, 10.4GT/s, 22 M Cache, Turbo, HT (150W) DDR4-2933
- iDRAC,Factory Generated Password
- iDRAC Group Manager, Enabled
- Chassis with up to 8 x 2.5" SAS/SATA Hard Drives for 1CPU Configuration
- PowerEdge 2U Standard Bezel
- Riser Config 1, 4 x8 slots
- 3200MT/s RDIMMs
- 32GB RDIMM, 3200MT/s, Dual Rank
- Amount: 6
- iDRAC9,Enterprise
- 1.92TB SSD SATA Read Intensive 6Gbps 512 2.5in Hot-plug AG Drive, 1 DWPD, 3504 TBW
- PERC H330 RAID Controller
- Minicard

KI has set up access requests (SSH) to the allocated VM for the COVIDX partners responsible for the deployment of Sandbox application.

5 Sandbox User Guide

This chapter provides guidance to administrators on how to install and configure the 1st release of the COVID-X Sandbox on top of physical or virtual servers. Moreover, it gives assistance to users who will be responsible for integrating their services into the Sandbox platform through the CI/CD Stack. It is aimed as reference documentation for joining third parties, being either solution providers to integrate their solution with the core services of the Sandbox or clinical partners wishing to locally deploy the Sandbox and ingest their health data sets.

5.1 Sandbox Installation and Configuration

This section presents the necessary steps that administrators should follow to install the Sandbox services. The minimal Sandbox deployment requires a multi-core virtual or physical server with 64GB of RAM, equipped with the necessary SSD Drives. The disk space has to be a multiple of the volume of the data that will be stored in the specific Sandbox deployment. Sandbox services can also be distributed to multiple hosts, as long as all hosts can communicate with each other. The software applications that must be installed to support the Sandbox deployment are listed below:

- Docker version >= 18.02
- Docker compose version >= 1.17
- Python >= 3.5

Detailed instructions of the installation process can be found in the respective COVID-X Gitlab repositories

5.1.1 Installation of the Data Management, Integration, and Harmonization services

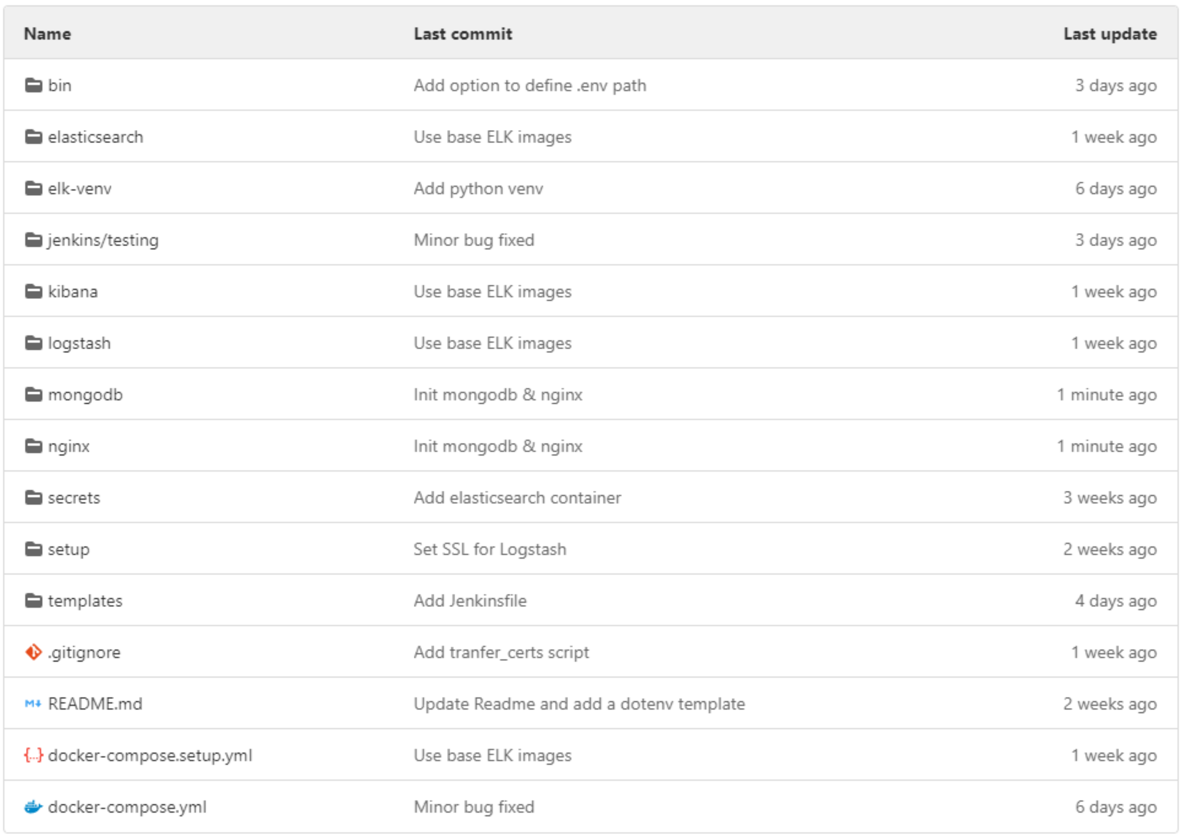
The Sandbox is capitalizing on the ELK Stack to support data management, integration, and harmonization services, including visualization services, as described in section 3.2. For this purpose, an Elasticsearch cluster is deployed, along with Kibana and Logstash. Moreover, Sandbox administrators may choose to install MongoDB that will be used to realize the Sandbox datalake for storing a massive amount of unstructured data.

An administrator must first download the source code from the COVID-X Gitlab repository to install the data management, integration, and harmonization services. Note that an invitation to the respective repository is required in order for someone to have access to the source code. On a Linux environment, the commands that must be executed are the following:

- **git clone https://gitlab.com/covid-x-prj/datamgmt/data-mgmt-harmonization.git**
- **cd data-mgmt-harmonization**

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

The root directory contains one subdirectory for each application that needs to be instantiated, as depicted in Figure 19. The installation process is based on Docker Compose for instantiating the necessary Docker containers and wrapper Shell scripts for configuring and preparing the environment.



Name	Last commit	Last update
bin	Add option to define .env path	3 days ago
elasticsearch	Use base ELK images	1 week ago
elk-venv	Add python venv	6 days ago
jenkins/testing	Minor bug fixed	3 days ago
kibana	Use base ELK images	1 week ago
logstash	Use base ELK images	1 week ago
mongodb	Init mongodb & nginx	1 minute ago
nginx	Init mongodb & nginx	1 minute ago
secrets	Add elasticsearch container	3 weeks ago
setup	Set SSL for Logstash	2 weeks ago
templates	Add Jenkinsfile	4 days ago
.gitignore	Add tranfer_certs script	1 week ago
README.md	Update Readme and add a dotenv template	2 weeks ago
docker-compose.setup.yml	Use base ELK images	1 week ago
docker-compose.yml	Minor bug fixed	6 days ago

FIGURE 19: DATA MANAGEMENT, INTEGRATION, AND HARMONIZATION REPOSITORY

Before starting the data management and harmonization services, the Sandbox administrator must create and configure the *.env* file. This configuration file defines various parameters of the system, such as the name of the cluster, the exposed network ports of the services, the admin password, etc. The full specification of the *.env* file is presented in Appendix 1.

To launch the data management, integration, and harmonization services, the administrator must execute *the bin/deploy.sh* script:

bash bin/bash.sh -k --kibana --logstash -- mongodb

This script utilizes the *docker-compose.yml* file to launch the defined Docker containers. The options passed to *deploy* script are the following:

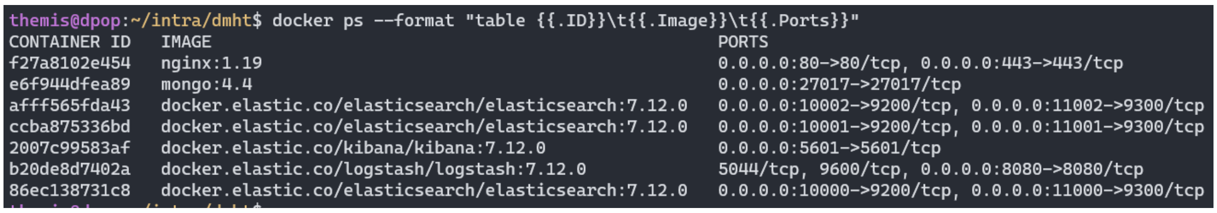
- **-k | --keystore:** Run *elasticsearch-keystore* to generate a new Keystore and *elasticsearch-certutil* to generate a new certificate under the *secrets* directory. Elasticsearch Keystore is used to store sensitive information and settings. The keystone will be created by default if there is not a Keystore under the secrets directory, even if this option is not enabled. You can use this option to replace a

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

previously created keystore or certificate. All the steps that are followed when enabling this option are described in detail in the next section.

- **--kibana:** Launch a Kibana service that will be connected to the instantiated Elasticsearch cluster. Kibana is configured based on the parameters in the .env file.
- **--logstash:** Launch a Logstash service that will be connected to the instantiated Elasticsearch cluster. Logstash is configured based on the parameters in the .env file.
- **--mongodb:** Launch the MongoDB database along with the ELK Stack services.
- **-h | --help:** Print a help message and quit.

If the deployment is successful, the docker ps command will list all the deployed containers realizing the data management, integration, and harmonization services. A similar output is depicted in Figure 20.



```
themis@dpop:~/intra/dmht$ docker ps --format "table {{.ID}}\t{{.Image}}\t{{.Ports}}"
CONTAINER ID   IMAGE                                                  PORTS
f27a8102e454   nginx:1.19                                             0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp
e6f944dfea89   mongo:4.4                                              0.0.0.0:27017->27017/tcp
afff565fda43   docker.elastic.co/elasticsearch/elasticsearch:7.12.0   0.0.0.0:10002->9200/tcp, 0.0.0.0:11002->9300/tcp
ccba875336bd   docker.elastic.co/elasticsearch/elasticsearch:7.12.0   0.0.0.0:10001->9200/tcp, 0.0.0.0:11001->9300/tcp
2007c99583af   docker.elastic.co/kibana/kibana:7.12.0                 0.0.0.0:5601->5601/tcp
b20de8d7402a   docker.elastic.co/logstash/logstash:7.12.0             5044/tcp, 9600/tcp, 0.0.0.0:8080->8080/tcp
86ec138731c8   docker.elastic.co/elasticsearch/elasticsearch:7.12.0   0.0.0.0:10000->9200/tcp, 0.0.0.0:11000->9300/tcp
```

FIGURE 20: DATA MANAGEMENT, INTEGRATION, AND HARMONIZATION CONTAINERS

The Sandbox administrator can create or modify the files under */logstash/pipelines* directory to modify the methods and sources that will be used for ingesting data into the system.

5.1.2 Configure Security Services

As described in section 3.3, the security services that are configured and integrated in the first release of the COVID-X Sandbox are part of the elastic X-Pack extension and include user authentication and authorization via a role-based access control system, authorized and encrypted inter-node communication in a multi-node cluster and encrypted client communication. For enabling the security services of the COVID-X Sandbox the basic X-Pack license is utilized, realizing all the aforementioned security features. In the configuration file of elasticsearch the following lines must be added:

xpack.license.self_generated.type: basic
xpack.security.enabled: true

The Sandbox administrator must execute the following three steps to generate the certificates that will be used for enabling encrypted communication among the multiple ndoes of the ELK cluster:

- Generating a certificate authority (CA): **elasticsearch-certutil ca**

- Generating and signing certificates for each node with the CA: **elasticsearch-certutil cert --ca elastic-stack-ca.p12**
- Copy the node certificate to the corresponding nodes of the cluster.

In order to generate certificates for securing inter- and intra-cluster communications the following steps are followed:

- Create a certificate authority for the Elasticsearch cluster: When generating a CA, a PKCS#12 keystore, elastic-stack-ca.p12, that contains the public certificate for the CA and the private key that is used to sign the certificates for each node, is created. Moreover, a password to protect the file containing the certificate and the key is defined. This file is retained along with its password in order to be able to add more certified nodes in the cluster. It is important to note that all nodes have a certificate that has been signed by the generated CA.
- Generate a certificate and private key for each node in the cluster: A PKCS#12 keystore, elastic-certificates.p12, is created for each that includes the node certificate, node key, and CA certificate. These files are also password protected and can be used by every node of the cluster.
- Attach the certificates to each node of the cluster: Each node certificate will be attached to the corresponding directory which is configured in the elasticsearch.yml of each node of the cluster as explained below for the case of Encrypting communications between nodes in a cluster and Encrypting HTTP Client communications.

The following snippet showcases the required commands needed to enable TLS and specify the information required to access the node's certificate in order to establish encrypted communications between nodes. We modify the **elasticsearch.yml** for each node of the cluster adding the following configurations:

```
server.ssl.enabled: true

elasticsearch.ssl.verificationMode: certificate\

xpack.security.transport.ssl.keystore.type: PKCS12

xpack.security.transport.ssl.keystore.path: /path/to/the/certificate.p12

xpack.security.transport.ssl.truststore.type: PKCS12

xpack.security.transport.ssl.truststore.path: /path/to/the/certificate_ca.p12
```

The above enables SSL and define the use of certificates and a certificate authority as the verification mode for our case. Furthermore, the type and the full path to the node certificate as well as the trusted keystore file are defined. Truststore is a trusted keystore file which includes the CA certificate which is used to sign the SSL/TLS certificates in the Elasticsearch cluster and it is also a PKCS12 file.

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

To enable encrypted HTTP Client communications, the following parameters must be added to the **elasticsearch.yml** file:

xpack.security.http.ssl.enabled: true

xpack.security.http.ssl.keystore.type: PKCS12

xpack.security.http.ssl.keystore.path: path/to/certificate.p12

xpack.security.http.ssl.truststore.type: PKCS12

xpack.security.http.ssl.truststore.path: path/to/certificate_ca.p12

Since the node's certificates are secured with a password, the administrator should also update the keystore and truststore files with the secure password:

elasticsearch-keystore add xpack.security.transport.ssl.keystore.secure_password

elasticsearch-keystore add xpack.security.transport.ssl.truststore.secure_password

By default, Kibana automatically detects whether to enable the security features based on the license and whether Elasticsearch security features are enabled. Thus, the administrator also needs to add the appropriate security commands for encrypted communications in the configuration file of kibana (kibana.yml):

server.ssl.enabled: true

elasticsearch.ssl.verificationMode: certificate

xpack.security.transport.ssl.keystore.type: PKCS12

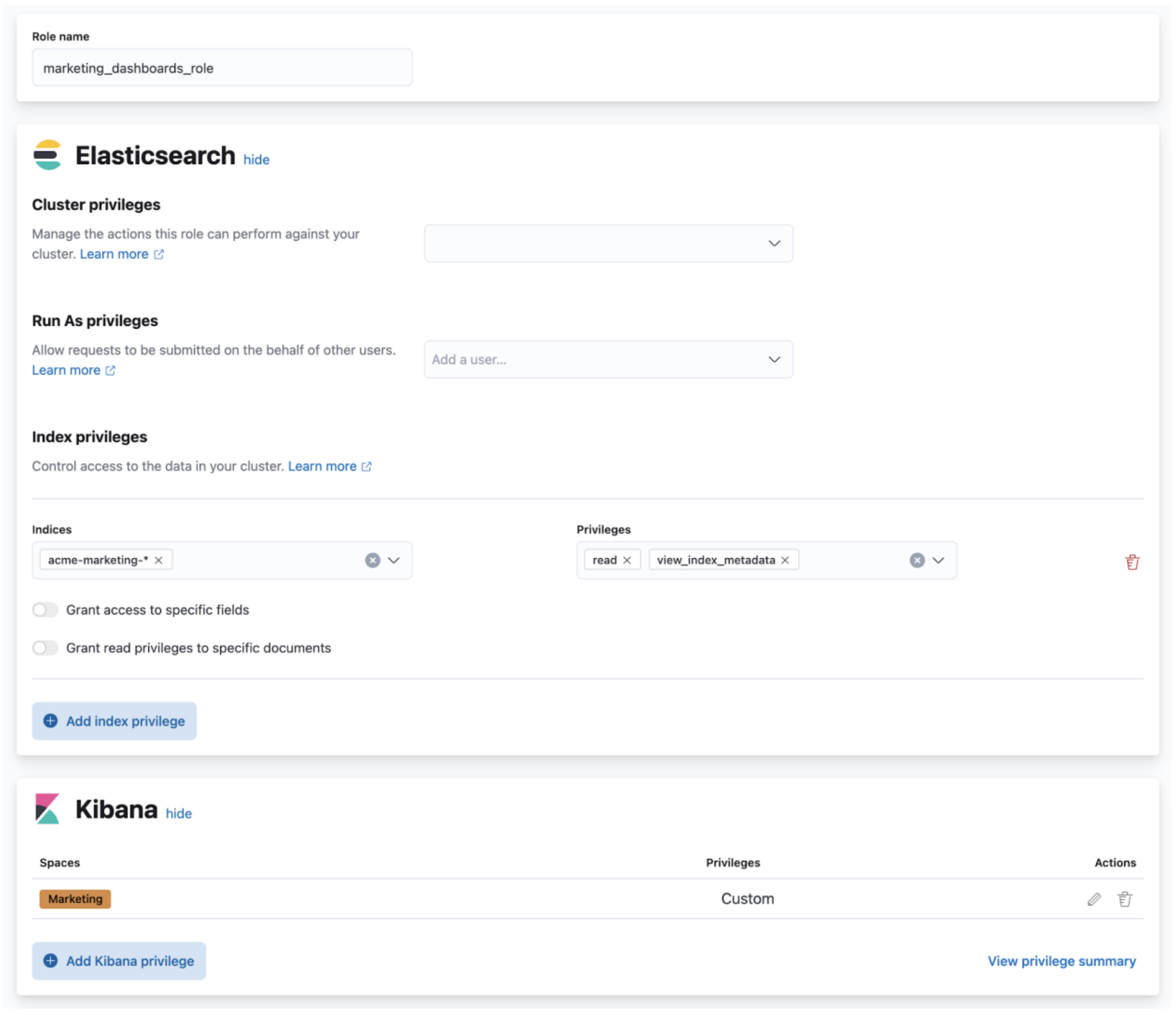
xpack.security.transport.ssl.keystore.path: /path/to/the/certificate.p12

xpack.security.transport.ssl.truststore.type: PKCS12

xpack.security.transport.ssl.truststore.path: /path/to/the/certificate.p12

Once the ELK Stack services have been deployed, following the instructions from the previous section, the Sandbox administrator can add roles with specific permissions and users. The roles can be assigned to one or multiple users. Kibana provides a useful web UI that for implementing these tasks, as shown in the figures below:

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



Role name

marketing_dashboards_role

Elasticsearch hide

Cluster privileges

Manage the actions this role can perform against your cluster. Learn more

Run As privileges

Allow requests to be submitted on the behalf of other users. Learn more

Add a user...

Index privileges

Control access to the data in your cluster. Learn more

Indices: acme-marketing-*

Privileges: read, view_index_metadata

Grant access to specific fields

Grant read privileges to specific documents

Add index privilege

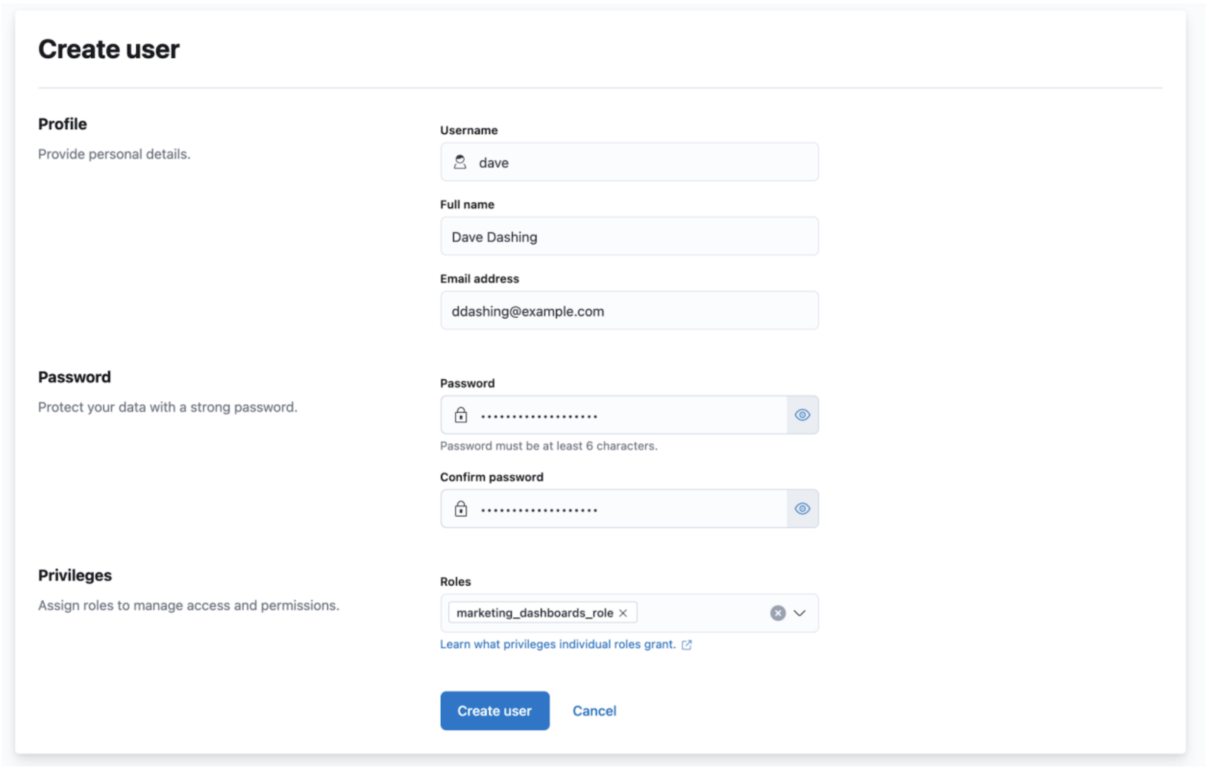
Kibana hide

Spaces	Privileges	Actions
Marketing	Custom	

Add Kibana privilege

View privilege summary

FIGURE 21: CREATE AN ELK ROLE



Create user

Profile
Provide personal details.

Username: dave
Full name: Dave Dashing
Email address: ddashing@example.com

Password
Protect your data with a strong password.

Password: ••••••••••••••••••••
Password must be at least 6 characters.
Confirm password: ••••••••••••••••••••

Privileges
Assign roles to manage access and permissions.

Roles: marketing_dashboards_role
Learn what privileges individual roles grant.

FIGURE 22: CREATE AN ELK USER

5.1.3 Installation of the API Gateway

The final task in the Sandbox installation process is the deployment of the API Gateway service. For this purpose, the Sandbox administrator must follow the steps listed below:

- Download the source code from the Gitlab repository: **git clone https://gitlab.com/covid-x-prj/sandboxapi/flaskapi.git** and **git clone https://gitlab.com/covid-x-prj/sandboxapi/kong.** These commands will download the kong and flaskAPI directories**.**
- In the kong directory, modify the **kong.yml** file, adding the following parameters:
  - Modify the Kong upstream, adding all the target endpoints on the instantiated nodes of the Elasticsearch cluster
  - For each Logstash HTTP pipeline, add a service that sends the required request to the adequate port, depending on the index.
- In both directories, modify the **docker-compose.yml** in the following parts:
  - Add the Docker network that is used for attaching the other services of the Sandbox.
  - Expose the desired ports. Note that the admin port should not be accessible externally.
- In both directories, execute the following command to deploy the API Gateway services: **docker-compose up -d**

5.2 CI/CD Guide

This section provides guidance to the Sandbox CI/CD pipeline that supports multiple branches in an entirely distributed environment by deploying software components over Docker. The Sandbox CI/CD has been described in detail in chapter 2. A template Jenkinsfile is included in Appendix 2.

5.2.1 Gitlab Repositories

A dedicated Gitlab group named "COVID-X-PRJ" has been created to host the source code of applications that will be integrated into the COVID-X platform. In order for partners to push their source code to the COVID-X repository, they must own a Gitlab account. Users can sign up in Gitlab using the following URL: https://gitlab.com/users/sign_up. Subsequently, a separate Gitlab project will be created for each partner under the COVID-X project. Partner must provide their email accounts associated with Gitlab to be invited to the respective Gitlab repository. An application should include the following files in order to be able to be deployed through the CI pipeline in a Docker container:

- Jenkinsfile: Define the Jenkins Pipeline as code.
- Dockerfile: File for building the Docker application.
- docker-compose.yml (optional): Deploy Dockerized application.

5.2.2 Jenkins

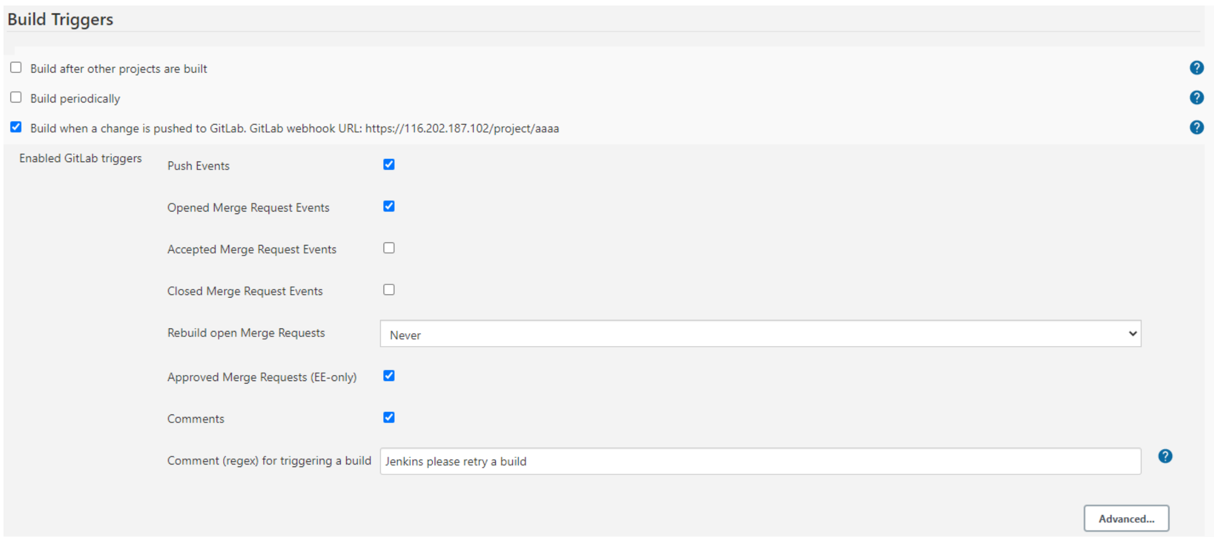
A Jenkins account will be provided to each partner with write permissions on specific workspaces. After that, a username/password will be communicated to them to start creating pipelines. Jenkins users can create pipelines following the steps:

1. **Create and configure a Jenkins pipeline**
   a. Create a ***New Item*** in Jenkins.
      - Enter an item name. It is recommended to give a name that matches the Gitlab project and the associated branch).
      - Select ***pipeline*** and click OK

   b. In the General tab, select ***Gitlab*** as a Gitlab Connection.

   c. The ***Build Triggers*** define when the pipeline runs. In the Build Triggers tab, select the options depicted in the following picture:

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

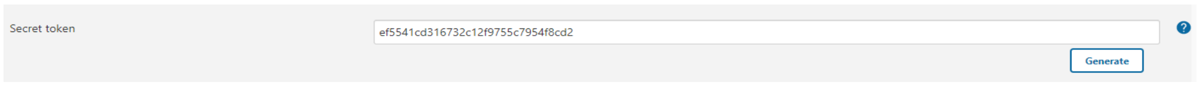

Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [x] Build when a change is pushed to GitLab. GitLab webhook URL: https://116.202.187.102/project/aaaa

Enabled GitLab triggers

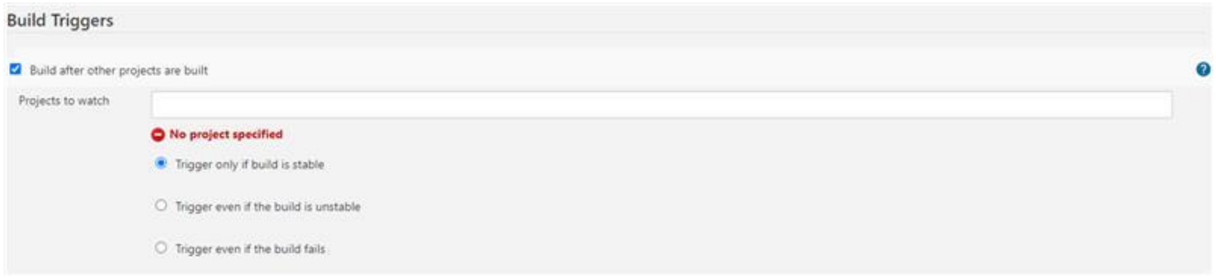
Push Events	✓
Opened Merge Request Events	✓
Accepted Merge Request Events	☐
Closed Merge Request Events	☐
Rebuild open Merge Requests	Never
Approved Merge Requests (EE-only)	✓
Comments	✓
Comment (regex) for triggering a build	Jenkins please retry a build

d. Click on the ***Advanced*** option and generate a ***Secret Token***. Keep the token as it will be used for connecting the Gitlab repository with the pipeline.


Secret token: ef5541cd316732c12f9755c7954f8cd2

e. Keep the ***GitLab webhook URL*** on the option title, as it will be used for connecting the Gitlab repository with the pipeline.

f. There is the option to launch a job after another project is finished. In such a scenario, where the job needs to be launched after another job is finished (ex: integration test-jobs, acceptance test jobs, etc.), the following option needs to be checked, referencing the job that precedes:


Build Triggers

- [x] Build after other projects are built

Projects to watch:

No project specified

- [✓ Trigger only if build is stable]
- Trigger even if the build is unstable
- Trigger even if the build fails

g. In the ***Pipeline tab***:

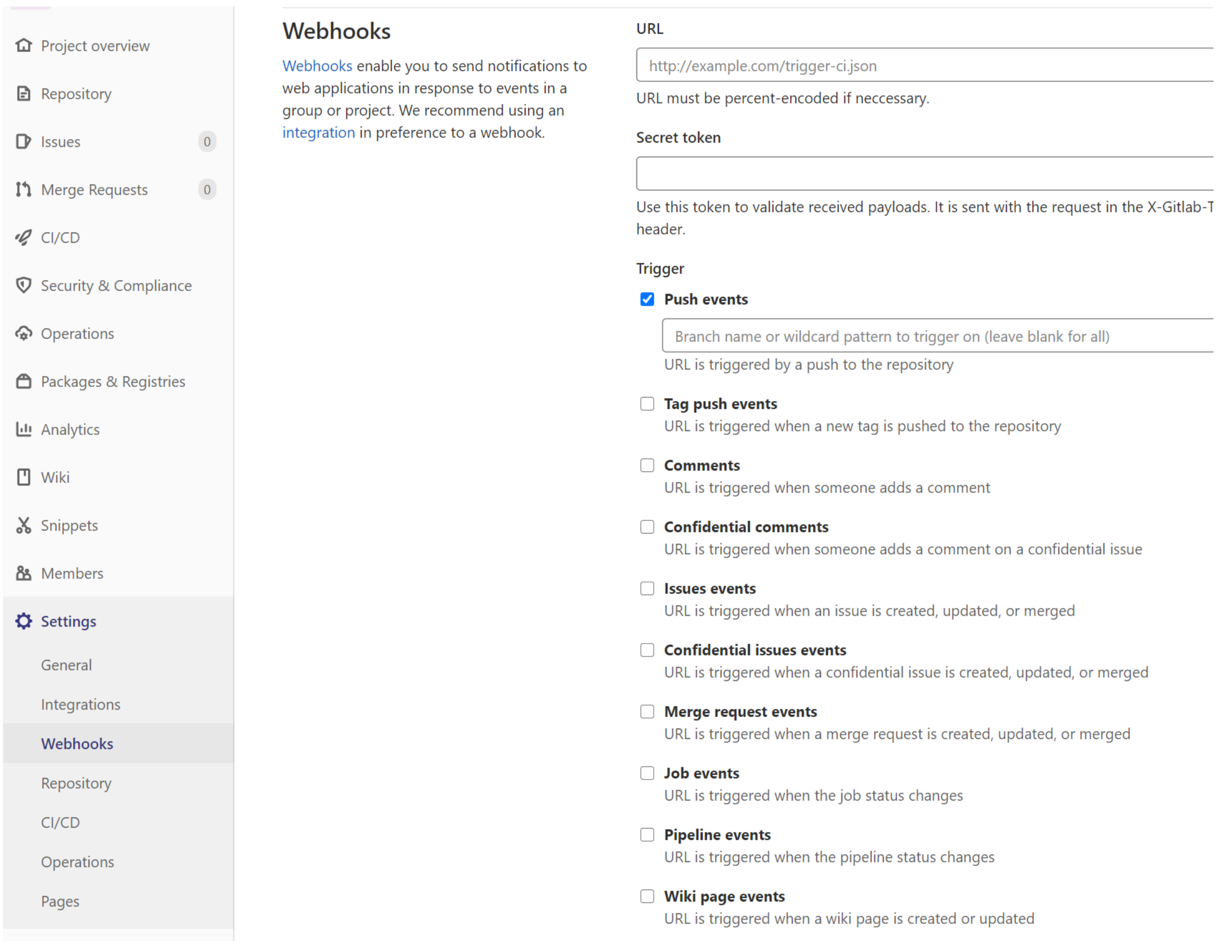
- Definition: Pipeline script from SCM
- Add the Repository URL (e.g. https://gitlab.com/covid-x-prj/<my-software-module>.git)
- Add your Gitlab credentials (username/password)
- In "Branches to build", add either branch (e.g. */master,*/development, etc.)
- Add Script Path (e.g., Jenkinsfile)

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

h. Save the configuration.

2. Create a Gitlab webhook to trigger the pipeline:

a. On the Gitlab repository that needs to be connected to the created Jenkins pipeline, navigate to Menu > Settings > Webhook page:



Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an integration in preference to a webhook.

URL

http://example.com/trigger-ci.json

URL must be percent-encoded if neccessary.

Secret token

Use this token to validate received payloads. It is sent with the request in the X-Gitlab-T header.

Trigger

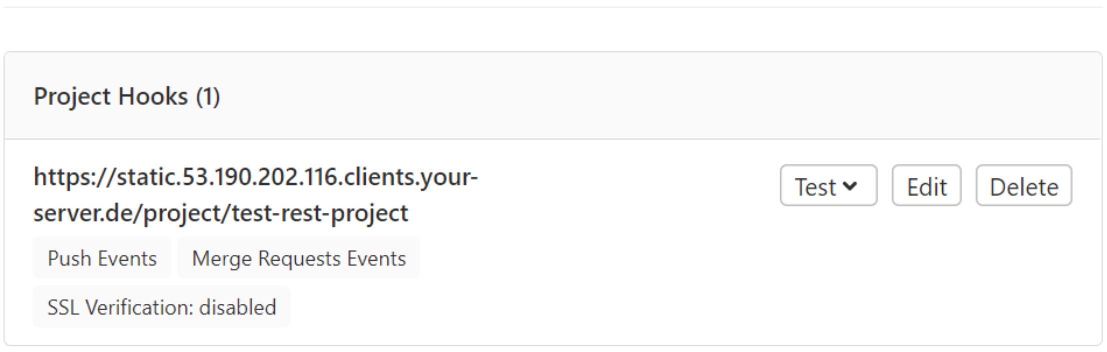
- [x] **Push events**
  Branch name or wildcard pattern to trigger on (leave blank for all)
  URL is triggered by a push to the repository
- [ ] **Tag push events**
  URL is triggered when a new tag is pushed to the repository
- [ ] **Comments**
  URL is triggered when someone adds a comment
- [ ] **Confidential comments**
  URL is triggered when someone adds a comment on a confidential issue
- [ ] **Issues events**
  URL is triggered when an issue is created, updated, or merged
- [ ] **Confidential issues events**
  URL is triggered when a confidential issue is created, updated, or merged
- [ ] **Merge request events**
  URL is triggered when a merge request is created, updated, or merged
- [ ] **Job events**
  URL is triggered when the job status changes
- [ ] **Pipeline events**
  URL is triggered when the pipeline status changes
- [ ] **Wiki page events**
  URL is triggered when a wiki page is created or updated

b. Fill the URL and Secret Token fields with the ones produced in the Jenkins pipeline.

c. Select the Giltab events that you want to trigger a new build of the pipeline.

d. Select "Add the webhook" when you are done. A new project hook appears at the bottom of the page. You can run a test to check that the pipeline is successfully launched.

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



Project Hooks (1)

https://static.53.190.202.116.clients.your-server.de/project/test-rest-project

Push Events

Merge Requests Events

SSL Verification: disabled

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

6 Conclusions

This deliverable has presented the activities related to the implementation of the first release of the COVID-X Sandbox, describing the internal services that have been developed for realizing the data management, integration, harmonization, and visualization functionalities, the security and API gateway services, as well as the components comprising the Sandbox CI/CD Stack to facilitate both the Sandbox services integration and testing but also the integration of third party solutions software. The development of the internal Sandbox services has followed the Service-Oriented Architecture approach, described in D2.1, as well as the best practices of the CI/CD process. By adopting this approach, development teams of the COVID-X project were able to develop and deploy the necessary services independently and therefore managed to introduce functionality to the system simultaneously with minimal dependency between them.

The deliverable has also presented the centralized cloud-based Sandbox deployment. Furthermore, it defines the requirements and the process that must be followed for deploying the Sandbox services at the local infrastructure of a clinical partner/data provider, to account for relevant emerging needs.

The first release of the Sandbox will be used for integrating and interconnecting the applications and components provided by the successful third-party SMEs that have joined the project during the 1st Open Call. This process will be described in deliverable D2.4 - First Report on interconnection of third parties, CI/CD and policies due month 12 of the project. The second and final release of the Sandbox will be described in D2.3 - Final Sandbox implementation and services provision due month 12.

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

7 References

[1] COVID-X Consortium, “D2.1 – Sandbox design & Datalake creation and ingestion,” 2021.

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

8 Appendix A: Sandbox dotenv Configuration File

This appendix presents the **.env** configuration file that defines the environmental variables used for deploying the Sandbox. For every variable, there is a default value that is set if not explicitly defined.

```
# Variables that will be used for deploying the ELK Stack
# Fil the variable that you want to define or elase the default value will
be used/
# Use: VARIABLE_NAME=VALUE

# ***** Hosts and Ports of the initial ELK Stack *****
# ***************************************************
# Thse variables will match the values defined in the docker-compose.yml
file
# They can be used for configuring pipelines and other internal ELK
processes
# Default values:
# ELASTIC_INIT_MASTER_NODE=dmht-elasticsearch
# ELASTICSEARCH_REQUEST_PORT=9200
# ELASTICSEARCH_CLUSTER_PORT=9300
#
# KIBANA_HOST=dmht-kibana
# KIBANA_PORT=5601
#
# LOGSTASH_HOST=dmht-logstash
# LOGSTASH_PORT=8080

# ELASTIC_INIT_MASTER_NODE=dmht-elasticsearch
# ELASTICSEARCH_REQUEST_PORT=9200
# ELASTICSEARCH_CLUSTER_PORT=9300

# KIBANA_HOST=dmht-kibana
# KIBANA_PORT=5601

# LOGSTASH_HOST=dmht-logstash
# LOGSTASH_PORT=8080

# ***** Hostnames of master eligble elasticsearch instances  *****
# ***************************************************************
# This variable will be used for adding data-nodes to elasticsearch cluster
# The value must match the resolvable hostname/IP of a cluster's master
eligible node
# Default value: ELASTIC_DISCOVERY_SEEDS=dmht-elasticsearch

# ELASTIC_DISCOVERY_SEEDS=dmht-elasticsearch

# ***** ELK Stack Resources *****
# ******************************
# The resources that will be reserved for the ELK applications
# Default values:
# ELASTICSEARCH_HEAP=1024m
# LOGSTASH_HEAP=512m

# ELASTICSEARCH_HEAP=1024m
# LOGSTASH_HEAP=512m

# ***** ELK Stack Version *****
# *****************************
# The version of the ELK services
# Default value: ELK_VERSION=7.12.0

# ELK_VERSION=7.12.0

# ***** Credentials *****
# ***********************
# Password for the admin user of Elasticsearch cluster (elastic).
# This is used to set the password at setup, and used by others to connect
to Elasticsearch at runtime.
# Default values:
# ELASTIC_PASSWORD=elastic

# ELASTIC_PASSWORD=elastic

# ***** Cluster Bootstrap Info *****
# **********************************
# General information about the ELK and Docker that will host the cluster
# Default values:
# ELASTIC_CLUSTER_NAME=ELK_Cluster
# ELASTIC_DOCKER_NETWORK=elk-net

# ELASTIC_CLUSTER_NAME=ELK_Cluster
# ELASTIC_DOCKER_NETWORK=elk-net

# ***** Elasticsearch Data Nodes *****
# ************************************
# The number of data nodes that will be created when launching the ELK
Stack
# More data nodes can be added later
# Default value ELASTIC_DATA_NODE_NUMBER=0

# ELASTIC_DATA_NODE_NUMBER=0
```

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

9 Appendix B: Example Jenkinsfile

This Appendix presents an example of a Jenkinsfile that defines the five main stages of the CI pipeline:

```
pipeline {
    agent any
    tools {
          maven 'Maven 3.6.3'
          jdk 'jdk8'
    }
    environment {
      APP_NAME = "test-module-rest"
      ARTIFACTORY_SERVER = "https://static.133.181.202.116.clients.your-
server.de:443/artifactory/cvregistry/"
      ARTIFACTORY_DOCKER_REGISTRY = "static.133.181.202.116.clients.your-
server.de:443/cvregistry/"
      BRANCH_NAME = "master"
      DOCKER_IMAGE_TAG = "$APP_NAME:R${env.BUILD_ID}"
      VM_DEV01 = "188.34.202.183:2376"
      VM_DEV02 = "157.90.24.51:2376"
      VM_DEV03 = "157.90.21.236:2376"
    }

    stages {
      stage('Checkout') {
          steps {
              echo 'Checkout SCM'
              checkout scm
              checkout([$class: 'GitSCM',
                        branches: [[name: env.BRANCH_NAME]],
                        extensions: [[$class: 'CleanBeforeCheckout']],
                        userRemoteConfigs: scm.userRemoteConfigs
              ])
            }
        }

        stage('Build with Maven') {
            steps {
                echo 'Build with Maven'
                sh 'mvn -f pom.xml clean package'
            }
        }

        stage('Build image') { // build and tag docker image
            steps {
                echo 'Starting to build docker image'
                script {
                    def dockerImage =
docker.build(ARTIFACTORY_DOCKER_REGISTRY + DOCKER_IMAGE_TAG)
                }
            }
        }

        stage ('Push image to Artifactory') {
            steps {
```

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

```
              withCredentials([[$class: 'UsernamePasswordMultiBinding',
credentialsId: 'jFrog_deployer_creds', usernameVariable: 'USERNAME',
passwordVariable: 'PASSWORD']]) {
                  echo 'Login to Artifactory Registry'
                  sh "docker login --password=${PASSWORD} --
username=${USERNAME} ${ARTIFACTORY_SERVER}"

                  echo 'Pull image with Build-ID'
                  sh 'docker push
"$ARTIFACTORY_DOCKER_REGISTRY$DOCKER_IMAGE_TAG"'

                  echo 'Logout from Registry'
                  sh 'docker logout $ARTIFACTORY_SERVER'
              }
            }
        }

      stage('Docker Remove Image from CI Server') {
        steps {
                sh 'docker rmi
"$ARTIFACTORY_DOCKER_REGISTRY$DOCKER_IMAGE_TAG"'
            }
        }

        stage('Deploy Image') {
          steps{
            script {
              docker.withServer("$VM_DEV01", 'dev_vm_pems') {
                withCredentials([[$class: 'UsernamePasswordMultiBinding',
credentialsId: 'jFrog_deployer_creds', usernameVariable: 'USERNAME',
passwordVariable: 'PASSWORD']]) {
                    echo 'Login to Artifactory Registry'
                    sh "docker login --password=${PASSWORD} --
username=${USERNAME} ${ARTIFACTORY_SERVER}"

                    echo 'Pull image with Build-ID'
                    sh 'docker pull
"$ARTIFACTORY_DOCKER_REGISTRY$DOCKER_IMAGE_TAG"'

                    echo 'Deploy image to VM'
                    sh 'docker run -d -p 8080:8080 --name "$APP_NAME"
"$ARTIFACTORY_DOCKER_REGISTRY$DOCKER_IMAGE_TAG"'

                    echo 'Logout from Registry'
                    sh 'docker logout $ARTIFACTORY_SERVER'
                }
              }
            }
          }
        }
    }
}
```