Tailored IoT & BigData Sandboxes and Testbeds for Smart, Autonomous and Personalized Services in the European Finance and Insurance Services Ecosystem

Infinitech

D6.4 – Tools and Techniques for Tailored Sandboxes and Management of Datasets - I

Revision Number	3.0
Task Reference	T6.2 - T6.3
Lead Beneficiary	HPE
Responsible	Alessandro Mamelli – Domenico Costantino
Partners	Participating partners in Task according to DOA
Deliverable Type	Report (R)
Dissemination Level	Public (PU)
Due Date	2020-07-31
Delivered Date	2020-09-30
Internal Reviewers	ASSEN, CCA
Quality Assurance	INNOV
Acceptance	WP Leader Accepted and Coordinator Accepted
EC Project Officer	Pierre-Paul Sondag
Programme	HORIZON 2020 - ICT-11-2018
	This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 856632

Contributing Partners

Partner Acronym	Role[1]	Author(s)[2]
HPE	Lead Beneficiary	Alessandro Mamelli – Domenico Costantino
LXS	Beneficiary	Pavlos Kranas
UBI	Beneficiary	Dimitrios Miltiadou
UPRC	Beneficiary	Georgios Makridis

Revision History

Version	Date	Partner(s)	Description
0.1	2020-03-31	HPE	ToC Version
0.2	2020-04-30	HPE	Updated ToC Version with requested contributions
0.3	2020-05-31	HPE and contrib. partners	1st draft contributions
0.4	2020-06-30	HPE and contrib. partners	2nd draft contributions
0.5	2020-07-31	HPE and contrib. partners	3rd draft contributions
0.6	2020-08-31	HPE and contrib. partners	4th draft contributions
1.0	2020-09-17	HPE	First Version for Internal Peer Review
2.0-2.1	2020-09-24-28	HPE - ASSEN – CCA - INNOV	Version for Quality Assurance - Review by the Quality Manager
3.0	2020-09-30	HPE	Version for Submission

[1] Lead Beneficiary, Contributor, Internal Reviewer, Quality Assurance

[2] Can be left void

Executive Summary

Within INFINITECH Work Package 6 “Tailored Sandboxes and Testbeds for Experimentation and Validation”, this document describes the results of Tasks T6.2 “Mechanisms and Tools for Tailored Sandboxes Provision and Configuration” and T6.3 “Integrated Management of Testbeds' Datasets” achieved during the first phase of the project, and provides the first version (D6.4) of this deliverable (out of the three foreseen in the whole of the work plan for WP6).

With respect to the general WP6 objectives, the deliverable mainly focuses on two of them, i.e.:

i. To provide tools and techniques for creating tailored sandboxes based on the selection of proper INFINITECH data assets, technological & regulatory building blocks.
ii. To provide a mechanism for integrated management of testbeds’ datasets, based on a continuous integration approach.

The achieved results provide key contributions for the fulfilment of the 1st major WP6 milestone (MS9 – First Version of ALL Sandboxes & Testbeds Available – foreseen for M16 of the project) and provides the initial release of the proposed tools and techniques.

The document is the accompanying textual specification of the other major deliverable result: the initial release of the proposed tools and techniques integrated and deployed into the INFINITECH blueprint reference testbed setup, designed and implemented upon one of the target INFINITECH infrastructures.

The document and the developed INFINITECH blueprint reference testbed setup constitutes the overall deliverable output.

The work has been carried out in close cooperation and coordination with the other INFINITECH WP6 tasks and work packages 2-3-4-5 tasks and partners, taking into account and integrating the delivered results and concepts (e.g. the INFINITECH Reference Architecture proposed by WP2) in a coherent and uniform manner.

The overall progress of such WP6 tasks will be one of the major drivers of the INFINITECH work package dedicated to the Large Pilots Operations and Stakeholders Evaluation of the proposed Financial and Insurance Services (WP7).

Index

List of Figures

List of Tables

Abbreviations/Acronyms

Table 1 Abbreviations

AWS	Amazon Web Services
AWS EBS	Elastic Block Store
AWS EKS	Elastic Kubernetes Service
AWS ELB	Elastic Load Balancer
AWS KMS	Key Management Service
CICD	Continuous Integration Continuous Development
CNI	Container Network Interface
DNS	Dynamic Name Resolution
HTAP	Hybrid Transactional and Analytical Processing
IAM	Identity and Access Management
K8s	Kubernetes
PoC	Proof of Concept
PV	Persistent Volume
PVC	Persistent Volume Claim
YAML	YAML Ain't Markup Language
VPC	Virtual Private Cloud

1 Introduction

INFINITECH is developing and validating BigData, IoT and Artificial Intelligence (AI) technologies for the finance and insurance sectors. In this direction, the project is advancing the state of the art in several technological development areas such as infrastructures for integrated, incremental and real-time analytics, semantic interoperability, as well as decentralized information sharing between stakeholders of the sector. Furthermore, the project designs and implements a range of pilots and use cases that aim at validating these technologies in real-life scenarios of the sectors.

Within INFINITECH, WP6 - Tailored Sandboxes and Testbeds for Experimentation and Validation – aims to achieve the following objectives:

i. To analyse the existing testbeds and specify their enhancements and upgrades.
ii. To provide tools and techniques for creating tailored sandboxes based on the selection of proper INFINITECH data assets, technological & regulatory building blocks.
iii. To provide a mechanism for integrated management of testbeds’ datasets, based on a continuous integration approach.
iv. To establish the 10+2 testbeds for experimentation and validation, including all relevant sandboxes.
v. To ensure continuous technical support for all testbeds, while establishing processes for certification/standardization of digital finance/insurance solutions.

1.1 Objective of the Deliverable

This document describes the preliminary results of INFINITECH WP6 Tasks T6.2 “Mechanisms and Tools for Tailored Sandboxes Provision and Configuration” and T6.3 “Integrated Management of Testbeds' Datasets” and provides the initial version (D6.4) of the deliverable (out of the three foreseen in the WP6 work plan). With respect to the general WP6 objectives mentioned before, this deliverable mainly focuses on objectives ii. and iii.

The results that have been achieved during the work provide key contributions for the fulfilment of the 1st major WP6 milestone (MS9 – First Version of ALL Sandboxes & Testbeds Available – foreseen for M16 of the project) and provide the initial release of the proposed tools and techniques.

The document is the companion textual specification of the other major result of the deliverable: the initial and preliminary release of the proposed tools and techniques, which have been also integrated and deployed into the INFINITECH blueprint reference testbed package, designed and implemented upon one of the target INFINITECH infrastructures.

The document and the implemented INFINITECH blueprint reference testbed setup constitute the overall deliverable output.

1.2 Insights from other Tasks and Deliverables

The following picture (Figure 1) depicts the interconnections between all the tasks inside the Work Package 6:

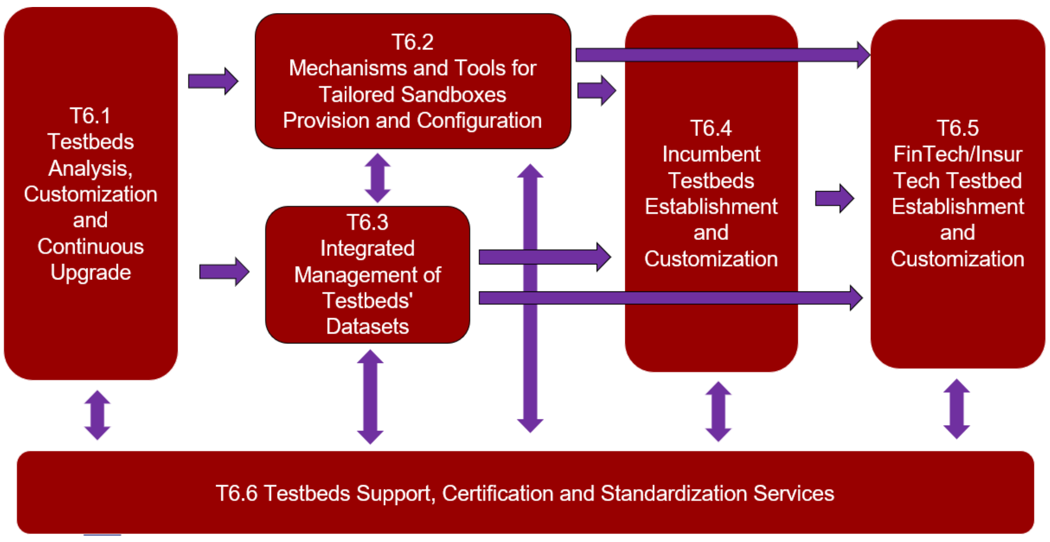
T6.1
Testbeds Analysis, Customization and Continuous Upgrade
T6.2
Mechanisms and Tools for Tailored Sandboxes Provision and Configuration
T6.3
Integrated Management of Testbeds' Datasets
T6.4
Incumbent Testbeds Establishment and Customization
T6.5
FinTech/InsurTech Testbed Establishment and Customization
T6.6 Testbeds Support, Certification and Standardization Services

Figure 1 Task dependencies in WP6

As stated before, the deliverable describes the preliminary results of INFINITECH WP6 Tasks T6.2 “Mechanisms and Tools for Tailored Sandboxes Provision and Configuration” and T6.3 “Integrated Management of Testbeds' Datasets”. An important input within WP6 for these tasks are the outcomes of Task 6.1 and its first available deliverable “D6.1 - Testbeds Status and Upgrades – I”, which contains the initial analysis for the current status of the existing partners infrastructure (hardware & software) that will be used as basis for all the testbeds to host the Pilots of the INFINITECH Project.

Moreover, the outcomes of T6.2 and T6.3 will feed the work of T6.4 and T6.5 for the deployment and customization in the target field infrastructure of the proposed tools and techniques for testbeds, sandboxes and management of testbeds datasets, and finally will also feed the work of T6.6 that will specify and implement processes for certifying and standardizing digital finance/insurance solutions in the project tailored sandboxes and testbeds.

Furthermore, looking to INFINITECH outside the WP6 border, the work done in T6.2 and T6.3 has been done with a strict and continuous collaboration and alignment with the INFINITECH work packages 2-3-4-5 tasks and partners, towards the integration of the delivered outcomes (e.g. the INFINITECH Reference Architecture proposed by WP2 or datasets management concepts by WP3) in a consistent way.

The comprehensive progress of such WP6 tasks will be one of the significant drivers of the INFINITECH work package focused to the Large Pilots Operations and Stakeholders Evaluation of the proposed Financial and Insurance Services (i.e. WP7).

1.3 Structure

The document consists of the following chapters:

- Chapter 2 “Relation to the general INFINITECH Reference Architecture” provides a summary of the INFINITECH Reference Architecture approach, and how the work done in the D6.4 deliverable relates to it in a coherent way
- Chapter 3 “Tools and techniques for Testbeds and Sandboxes” describes the approaches and technologies leveraged to implement the Testbeds and Sandboxes concepts within the INFINITECH project
- Chapter 4 “Tools and techniques for Management of Datasets ” describes the tools and techniques that leveraged for the management of datasets within the INFINITECH project
- Chapter 5 ”Overview of the INFINITECH blueprint reference testbed” describes the initial design and implementation of the INFINITECH blueprint reference testbed, through the actual realization (with a full compliance) of the INFINITECH Reference Architecture Development and Deployment views.

Moreover, the planned blueprint environment associated to the initial and preliminary Proof of Concept implementation of one of the official INFINITECH pilots is also described.

- Chapter 6 “Conclusions” summarizes the results of the work done in the deliverable and the next steps foreseen for the related tasks.
- Chapter 7 “Appendix A: Literature” provides details about all the cited work.

2 Relation to the general INFINITECH Reference Architecture

This chapter illustrates a summary of the INFINITECH Reference Architecture approach, described in depth in deliverable "INFINITECH-D2.13 - Reference Architecture - I", and how the work done in the D6.4 deliverable relates to it in a coherent way.

2.1 INFINITECH Reference Architecture

The INFINITECH Reference Architecture (IRA) leverages the "4+1" architectural view model as the methodology to cover all the aspects of the different problems the Project and the Consortium want to address.

The "4+1" architectural view model [1] is a methodology to design an architecture for a software platform, having the main capacity of describing it from 5 concurrent "views".

These views represent the different stakeholders who could deal with the platform and the architecture, from the management, development and user perspectives. The four main views which facilitate the definition and design of the architecture are the logical, process, development and physical ones, while the "+1" view is represented by the use cases or scenarios, thus making this model an abstraction of the developed solution/platform and the basis for the development.

In the following, the meaning of the different views is explained:

- **Logical view:** it represents the range of functionalities or services that the system provides to the end users, and can be shown as block diagrams.
- **Process view:** it represents the system processes and data flows and how the different processes and building blocks communicate between each other, with details on the runtime behaviour of the system.
- **Development view (or Implementation view):** it illustrates the software management aspects of the system, from the programmer's point of view.
- **Physical view (or Deployment view):** it describes the lower levels of the architecture, dealing with the physical infrastructures where the software components are deployed and run and the physical connections between them. Deployment diagrams can represent it.
- **Scenarios:** the architecture of the system can be explained from the end user's point of view too, with the description of use cases. The use cases or "scenarios" aim at describing some possible functioning situations of the system and interactions between components. The validation and assessment of functionalities are usually performed by this view.

The INFINITECH RA presented in Figure 2 defines *layers* as a way to logically group components.

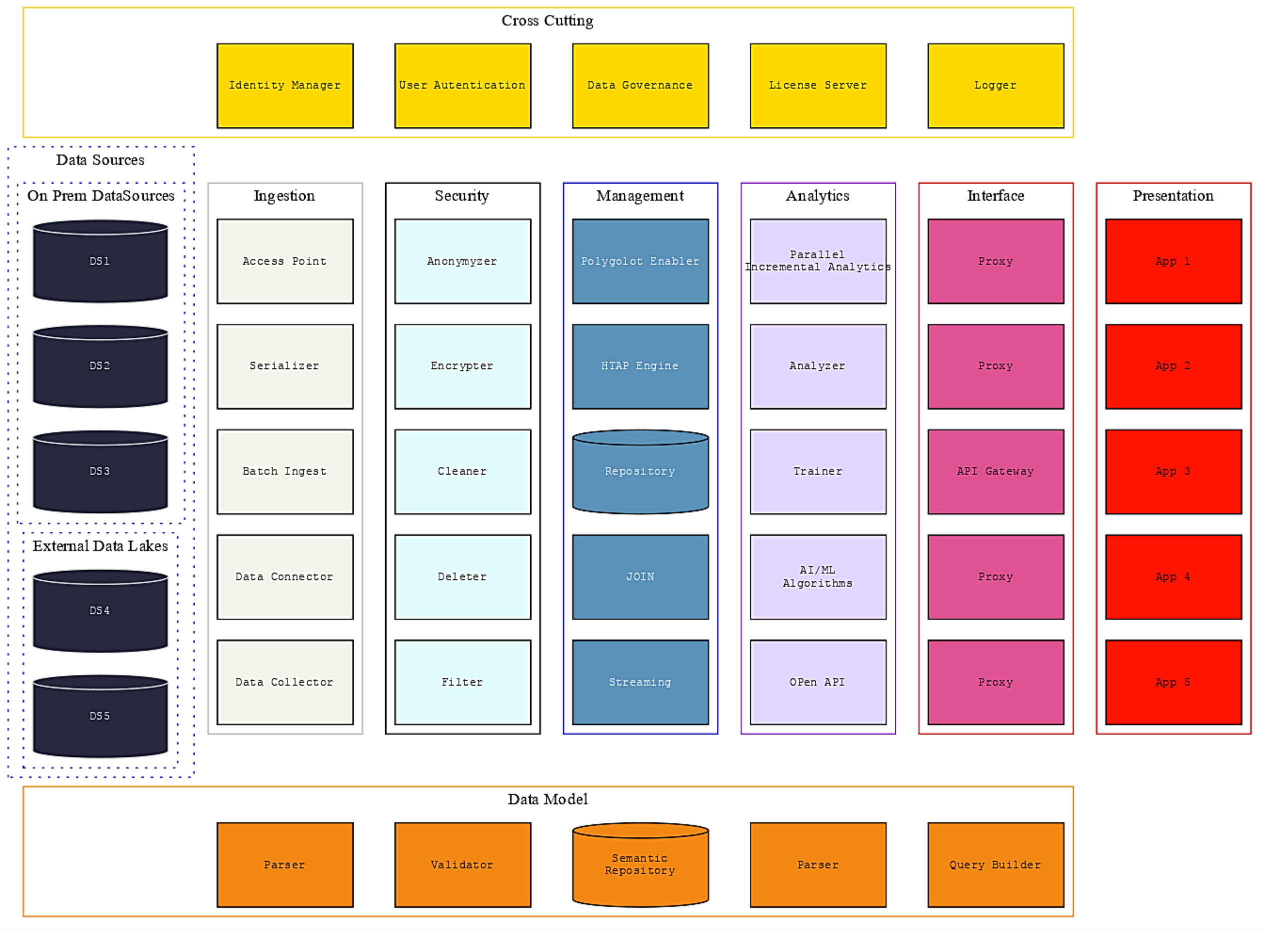

Figure 2 INFINITECH Reference Architecture logical view

The identified layers are:

- **Data Sources:** at the infrastructure level, there are the different sources of data (database management systems, data lakes holding non-structural data, etc.).
- **Ingestion:** a layer of data management usually associated with data import, semantic annotation and filtering from data sources.
- **Security:** a layer for management of the clearance of data for security, anonymization, cleaning of data before any further storage or processing.
- **Management:** a layer responsible for the data management aspects, including the persistent storage in the central repository and the data processing enabling advanced functionalities such as Hybrid Transactional and Analytical Processing (HTAP), polyglot capabilities, etc..
- **Analytics:** a layer for the AI/ML/DL components.
- **Interface:** a layer for the definition data to be produced for clients (e.g. user interfaces, etc.).
- **Cross Cutting:** a layer with service components that provide functionalities orthogonal to the data flows (e.g. Authentication, Authorization, Accounting, etc.).
- **Data Model:** a cross cutting layer for modelling and semantics of data in the data flow
- **Presentation/Visualization:** a layer usually associated with the presentation applications (desktop, mobile apps, dashboards and the like).

It should be noted that the IRA does not impose any pipelined or sequential composition of nodes. However, it is recommended to consider each different layer and the relative components to solve specific problems of the use case.

In the end, the major D6.4 document purpose, and its relation to IRA with a full alignment to its building blocks, is to describe in particular how the IRA Development and Physical views have been tackled and designed by the Consortium in terms of the concrete specification and realization of the fundamental and target INFINITECH concepts of Testbeds, Sandboxes and Datasets management, and related tools and techniques for their effective setup and deployment in the INFINITECH pilots and validation scenarios.

3 Tools and techniques for Testbeds and Sandboxes

This chapter describes the tools and techniques that will be leveraged to implement the testbeds and sandboxes concepts within the INFINITECH project, considering that the INFINITECH RA is designed leveraging a paradigm based on a **microservices** [2] architecture implementation, with services interacting among them through REST APIs.

Key pillars for the implementation and deployment of a microservices based architecture are the **containers** technology [3] and its leading open source containers orchestration solution, i.e. **Kubernetes** [4]. Therefore, in order to understand such methodological and technological choices for the realization of the INFINITECH RA, some key concepts related to containers, microservices and Kubernetes, and how their appearance in the IT environments has deeply changed the software development approaches, are presented.

Finally, the details about how the concrete usage of such technologies benefits and enables the definition of the INFINITECH testbeds and sandboxes concepts are also provided.

3.1 Containers benefits

In the last few years there has been a strong transformation, similar to what happened in the early 2000s with the advent of virtualization, due to containers spread which led to rethinking both how to manage the infrastructure and how to design and build the applications.

The advent and the diffusion of containers has made it possible to improve the computational management of infrastructures, thanks to the possibility of removing the overhead generated by the use of the hypervisor (integrated software that allows to virtualize the HW resources of a server and make them available among several applications), and through the usage of the functionalities already available within the Linux OS kernel, as in Figure 3.

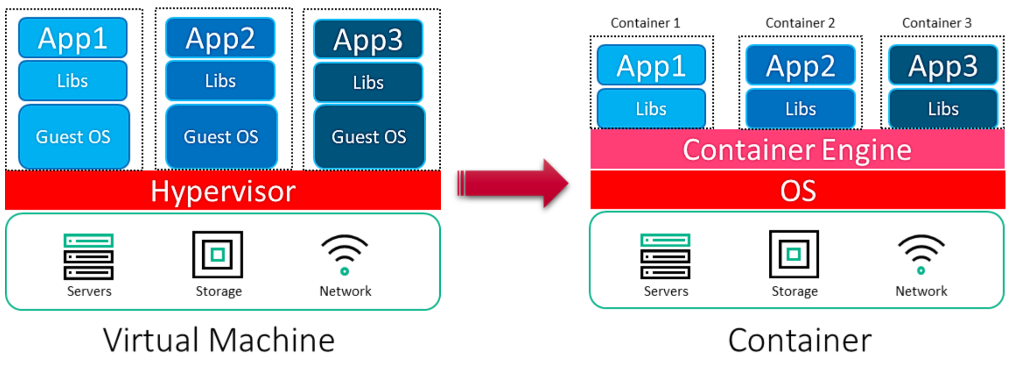

Figure 3 VM vs Container

The container engine, to date the most used one is Docker [5], takes advantage of

- **Linux-Control groups**: Allow each container to get its fair share of memory, CPU, disk I/O and Network stack. At the same time a single container cannot bring the system down by exhausting one of those resources.
- **Linux-Namespaces:** Provide the possibility to isolate the processes running into a container from processes running in another container or in the host system.

to run containers like VMs isolated each other.

In the end, a container is a package that contains code, system libraries, dependencies and software tools.

An example is given in Figure 4.

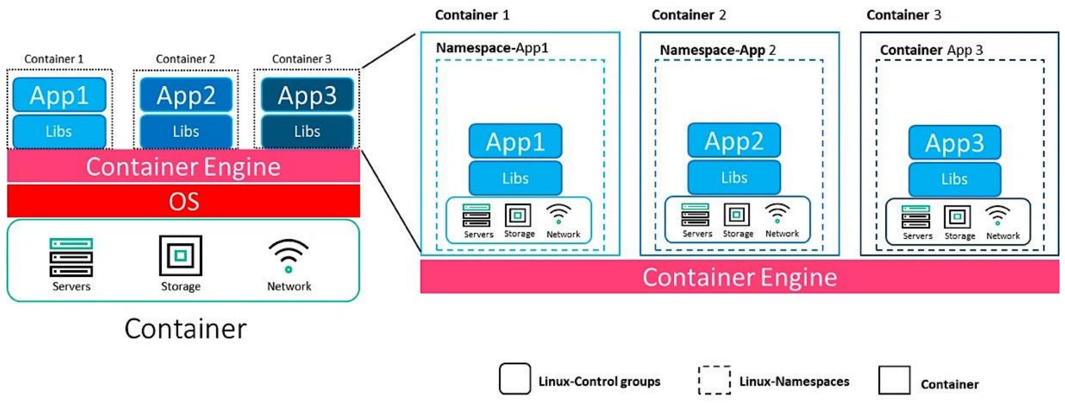

Figure 4 Container kernel properties

The main advantages to use containers respect VMs can be summarized in these macro points:

- **Size**: a Container is small.
- **Overhead**: no fully OS is required.
- **Speed**: Boot time is faster.
- **Scaling**: Real time provisioning.

At the same time, in order to take advantage of containers is necessary to rethink and redesign the currently monolithic applications into microservices based applications.

3.2 Microservices approach

The spread of the containers led, as already mentioned, to the necessity to change the approach of the developers in the creation of an application, moving from design of monolithic applications, where the various components (generally UI, Business logic and Data-layer) were strongly coupled among them, to microservices applications where the various components (i.e. microservices) are decoupled from each other (see Figure 5)

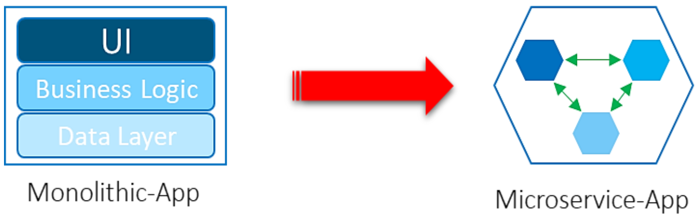

Figure 5 Monolithic vs Microservice

This methodological change has both advantages and disadvantages.

The advantages can be summarized as:

- **Simple to develop**: Each microservice is independent and small.
- **Simple to upgrade**: Since each microservice is independent it's possible to upgrade each component independently.
- **Simple interaction**: Each microservice communicates with each other through well-defined and standard interfaces (API).
- **Simple to scale**: Each microservice can be scaled independently.
- **Scale on demand**: It's possible to run multiple microservices behind a load balancer to scale on demand request.

The disadvantages can be summarized as:

- **Complexity**: Splitting an application into multiple independent microservices increases the complexity of the deployment process.
- **Monitoring**: Monitoring each microservice requires to have many metrics and logs to manage it.
- **Performance**: All communications occur on network so they are slower than memory communications.
- **Debugging**: a Monolithic application is much easier to debug and test due to the fact it is composed by single indivisible units.

Our approach for a rapid iterative development of a microservice based software infrastructure builds upon a widely used methodology like **DevOps**, as described in section 5.1.

3.3 Kubernetes containers orchestration

The arrival on the market of containers and related microservices based applications, on the one hand enabled applications that quickly scale according to the requirements and that could be easily updated, on the other hand meant that software previously managed as a single indivisible piece was split into several dozens of microservices (containers), making it more difficult to manage them.

In this context, the necessity to develop a tool that was able to manage the life-cycle of the microservices (deployment, scaling, and management) arose: such tool was developed by Google with the name of "Project Seven of Nine "and released as open source software in 2014. Today such tool is widely known as **Kubernetes**.

Kubernetes provides:

- **Service discovery and load balancing**
  Kubernetes can expose a container using the DNS name or using its own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.
- **Storage orchestration**
  Kubernetes allows to automatically mount a storage system of different types, such as local storages, public cloud providers, and more.
- **Automated rollouts and rollbacks**
  You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new containers.
- **Automatic bin packing**
  You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
- **Self-healing**
  Kubernetes restarts containers that fail, replaces containers, kills containers that do not respond to your user-defined health check, and does not advertise them to clients until they are ready to serve.
- **Secret and configuration management**
  Kubernetes lets you to store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

3.3.1 Kubernetes architecture

A Kubernetes (aka **K8s**) cluster is made up of two macro blocks, the first one called Control Plane (Master) and the second one called Data Plane (Worker).

The Control Plane constitutes the brain of the cluster and internally it is made up of the following components:

- **Kube-APIserver** is the component that exposes cluster API and truly is the main component, since Kubernetes has been designed and built to base all the operations on the use of the API.
- **Etcd** is a key value database that maintains all information relating to the status of the cluster.
- **Kube-scheduler** schedules on which nodes of the Data Plane runs the containers (in Kubernetes named POD, the smallest deployable units of computing that can be created and managed in Kubernetes) based on the resources required, cluster status and affinity and anti-affinity rules.
- **Kube-controllermanager** consists of a set of control processes which:
  - Check if cluster nodes are active.
  - Check if number and status of running POD it's required one.
  - Control and create token to access on the K8s resource.
  - Populates the Endpoints object (that is, joins Services & Pods).

The Data Plane is the part where the workload is carried out, i.e. where the PODs are put into execution and it is characterized by the following components:

- **Kube-proxy** is a proxy that allows communication to the PODs from within and outside the cluster.
- **Kubelet** checks that PODs are running.
- **Container runtime (engine)** is a software responsible for running containers in Kubernetes can be used(Docker, CRI-O and containerd).

In the Figure 6 the Kubernetes Cluster components is depicted:

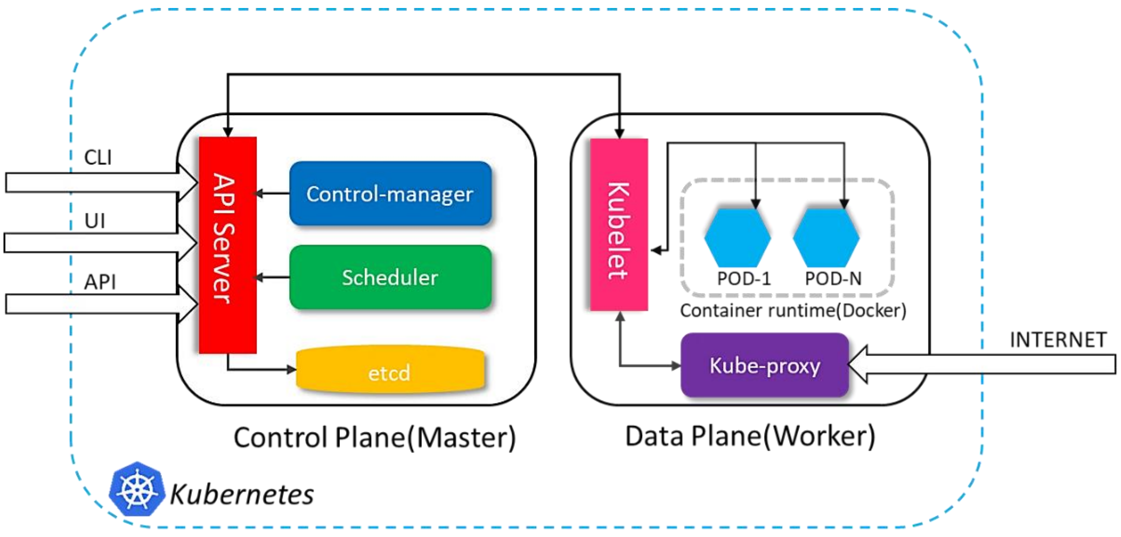

Figure 6 Kubernetes Architecture

For an outline of the entire Kubernetes solution, see the documentation on kubernetes.io.

However, for the purpose of this chapter it is sufficient to now outline two main concepts available in Kubernetes that we will use later to implement the Sandbox concept.

1. **Namespaces**: They are a logical grouping of a set of Kubernetes objects to whom it's possible to apply some policies, in particular:
   - **Quote** sets the limits on how many HW resources can be consumed by all objects.
   - **Network** defines if the namespace can be accessed or can access to other Namespaces, in other word if the Namespace is isolated or accessible.

Different policies can be given to different namespaces.

2. **POD** (Figure 7) is the simplest unit in the Kubernetes object. A Pod encapsulates one container, but in some cases (when the application is complex) a POD can encapsulate more than one container. Each POD has its own storage resources, a unique network IP, access port and options related to how the container/s should run.

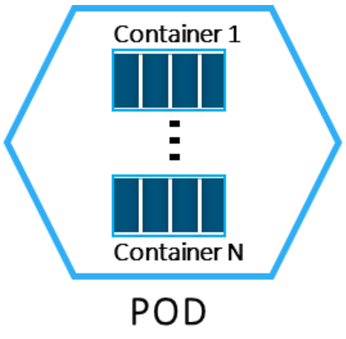

Figure 7 Kubernetes POD

3.4 INFINITECH Testbeds

At the time of writing and in alignment with the content reported in the major current outcome of WP6 task T6.1 (see section 1.2), i.e. the deliverable "D6.1 - Testbeds Status and Upgrades - I", INFINITECH will make available a number of testbeds for experimentation, testing and validation of BigData, AI and IoT solutions, including:

- 10 testbeds that are established in the data centres of incumbent financial organizations (on premise testbeds).
- 1 testbed that will be provisioned and established in order to support the experimentation of the FinTech and InsurTech pilots and enterprises of the consortium, hosted on the partner NOVA's Data Center.
- 1 testbed that will be provisioned and established in order to support the experimentation of the INFINITECH blueprint reference testbed (see Section 4.2 of the deliverable, associated to one of the official INFINITECH pilots), hosted on the AWS (Amazon Web Services) [6] public provider.

Accordingly, the current, (at the time of writing, subject to possible evolutions along the project lifecycle), INFINITECH project plan is to deliver 15 pilots: 10 out of 15 will be carried out on dedicated on premise Data Centres, while the remaining 5 out of 15 will be carried out on the NOVA's Data Centre, a shared INFINITECH Data Centre. In addition, a blueprint reference testbed will be also provided, built upon the requirements of one of the INFINITECH pilots, as stated above.

The set of hardware resources like storage, compute and network will be considered a **testbed**, as shown in Figure 8.

Servers
Storage
Network
Data Center=TESTBED

Figure 8 Testbed

It is not relevant where these resources are deployed: they can be inside a private Data Centre or in any cloud provider.

Therefore, the 15 pilots that have been foreseen will be executed in 10+2 testbeds, in addition to the blueprint reference testbed, as shown in Figure 9.

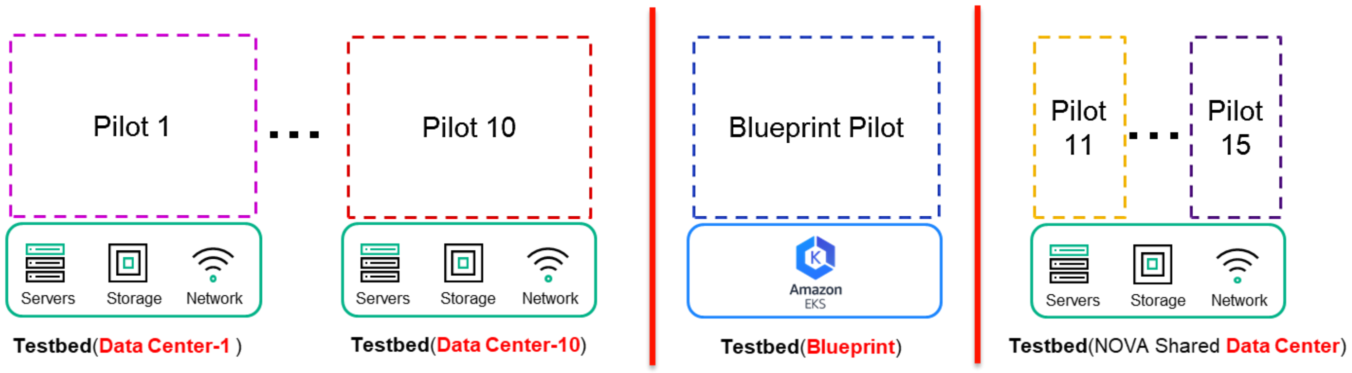

Figure 9 Testbeds and Pilots

3.5 INFINITECH Sandboxes

Each INFINITECH pilot will have one or more Use Cases (realized by one or more pilot Apps, each one realized by one or more INFINITECH microservices): in our vision each Use Case will be a **Sandbox** provisioned by the leverage of Kubernetes Namespaces.

In fact, as we already said in the previous paragraph, the Kubernetes Namespace feature makes it possible to logically isolate the objects (mainly PODs) inside it from other Namespaces. Therefore, each **dedicated Testbed** will only have one Kubernetes cluster with as many Namespaces as the number of Use Cases to be implemented for a single pilot (see Figure 10). In the other case, each (2 out 10) **shared Testbed** will have one Kubernetes cluster for each pilot it has to host and manage (see Figure 11).

In the end, in this general context, each pilot App will be realized as a Kubernetes POD.

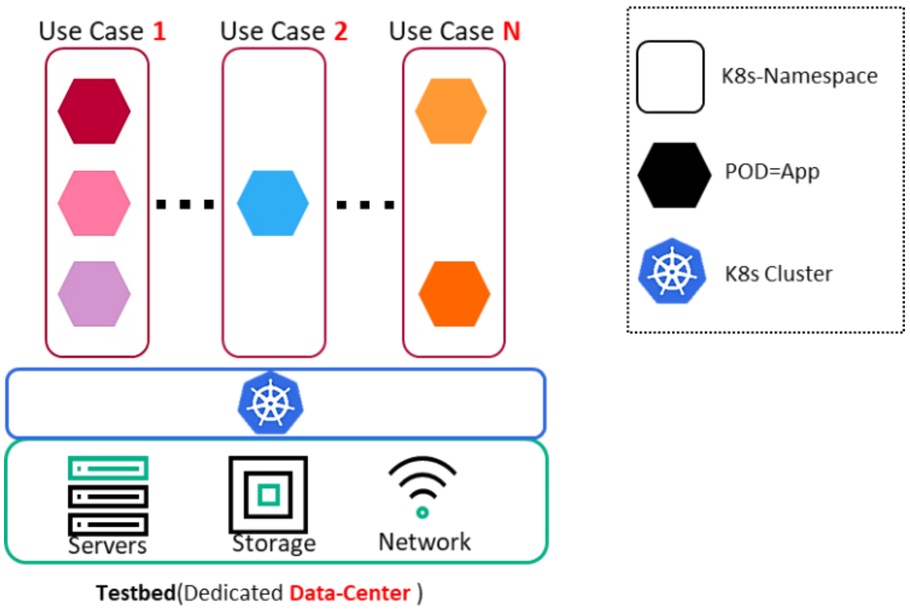

Figure 10 Sandboxes in a dedicated Testbed

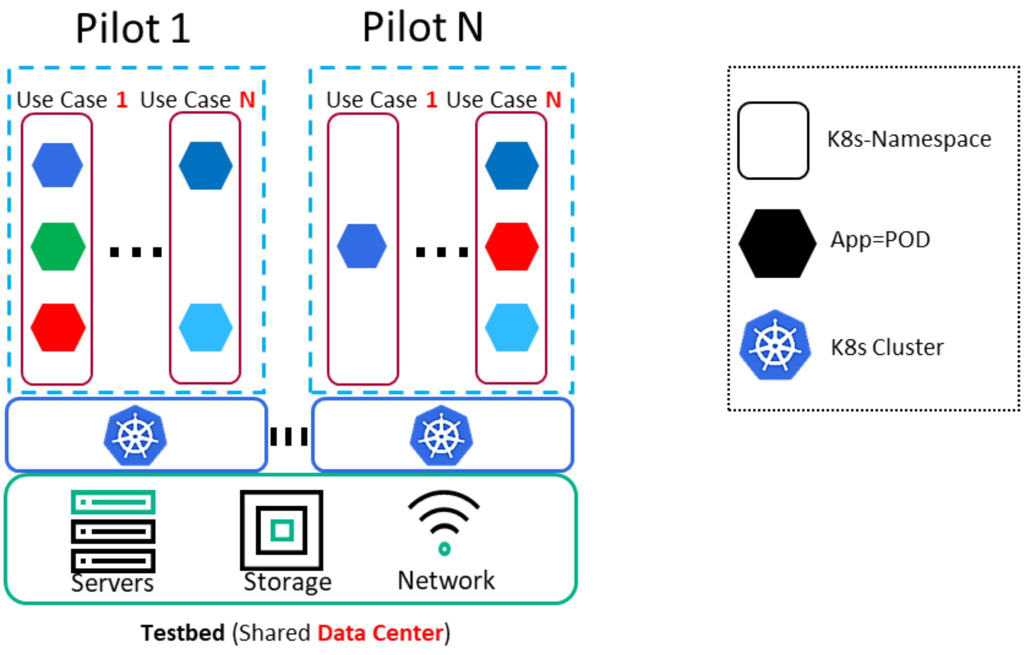

Figure 11 Sandboxes in a shared Testbed

4 Tools and techniques for Management of Datasets

This chapter describes the tools and techniques that will be leveraged for the management of datasets within the INFINITECH project and how they are linked and mapped with the concepts and techniques related to testbeds and sandboxes (see Section 3) and also with respect to the blueprint reference testbed environment (see Section 4.2).

4.1 Data Sources and Data Access

The INFINITECH platform aims to host and serve the needs of a variety of applications and tools used by the finance and insurance sectors, which have diverse needs and requirements for data access. We envisage that different types of data sources will be supported such as:

- structural, semi-structural or completely un-structured data,
- static data, often called data at-rest or streaming data, while the source might be stored on premise, in a third-party organization or should be imported inside the deployed sandbox. For instance, a common solution might need to use historical data in order to perform a post processing and apply ML/DL analytics for risk assessment.

However, modern enterprises are not only interested in extracting knowledge and identifying risks or opportunities based on historical and often obsolete data, but they also need to extract this type of information from real data, which introduces a significant challenge regarding data management from the tailored sandbox perspective. To further complicate the needs of the platform, other requirements coming from the finance and insurance sectors are the identification of potential fraudulent financial transactions as they occur, or pre-processing of data from IoT devices as it arrives. Data analysts often tend to use different datasets that can be retrieved from third party organizations or other external sources such as social media to enrich their algorithms with deeper knowledge in order to build a more accurate profile of their customers, either referring to individuals or to enterprises. Finally, additional considerations must be taken into account when dealing with the volume and privacy of the data of a finance organization, where moving data from the source to a sandbox might not be feasible either due to regulatory constraints, or simply due to the overall volume of the datasets.

In order to address all these requirements, the INFINITECH RA has been designed in order to take into account the different types of data sources. With respect to the implementation to support the functionality needed for data management and processing, details have been given in the corresponding deliverables of WP3, WP4 and WP5.

In looking at the techniques and tools that will allow the tailored sandboxes to enable the data management of these diverse environments, the focus is on data access. In other words, how the different components that will be deployed inside the sandbox can be integrated in a way that allows them to make use of all the aforementioned data sources.

We can identify 6 different focuses for data access:

- Static data ingestion
- Dynamic data ingestion
- IoT streaming
- Direct access to on premise data sources
- Blockchain data access
- Third party data access

In the following subsections, we will describe the details of each of those different means of data access and will analyse the requirements and technical considerations that need to be tackled by the infrastructure, when orchestrating the deployment and maintenance of the integrated solution. The next section will give more insights on how the INFINITECH platform overcomes the barriers and deals with the challenge of securely allowing those diverse means for data access, thus enabling the automated deployment of the integrated solution.

4.1.1 Static data ingestion

This is the most typical case for data access that is foreseen to be a requirement for the majority of the cases that will need to be deployed into an INFINITECH sandbox. The data analyst has their own data stored into a database management system or other form of persistent storage, and needs to extract a specific dataset and ingest it into the central data repository deployed inside the sandbox so that they can use the tools provided by the platform to perform the analysis. They will have to extract this information in a static file in a specific format (i.e. csv) and trigger the data ingestion process of INFINITECH. We call this data access mode static data ingestion, as the file will be created once, and will be used in each deployment. This access mode also covers the needs for a data analyst to make use of an existing synthetic dataset that has been provided and made available to the platform, via the INFINITECH marketplace. The generated file must be stored into a persistent storage volume and the latter must be visible from the sandbox so that the tools deployed inside can have access to it. This mainly refers to the data ingestion component that will take care of the data migration process, possibly applying a pre-processing algorithm to clean, harmonize and anonymize the dataset and finally store the raw data into the data repository. It will also support cases where direct access to the static file might be needed by the analytical tools on the analytics layer of the RA, where there is no need for additional processing by the lower logical levels of the architecture (i.e. a spark job needs to grab everything from a csv file, where no additional processing like a filter or join operation can be pushed down to the processing layers).

4.1.2 Dynamic data ingestion

In this scenario, data ingestion does not take place once in the initialization of the sandbox, rather the data might be migrated periodically to the data repository. We call this dynamic data ingestion as the dataset inside the sandbox will be updated with new data after a given period of time. This will cover cases where the requirement is to have the sandbox deployed, integrated and synchronised with the data sources of the organisation. For instance, the integrated solution might need to get updated with the current snapshot of the data, so that the data analyst can benefit from a daily picture of the transactions of any given customer. Rather than having a human intervene to manually extract the data needed into a static file that will be loaded into the system, the data provider can implement specific APIs so that the data migration can be done in a fully automated manner every time (i.e. end of the business day). The data ingestion component of INFINITECH will need to open a connection to those APIs in order to retrieve the data according to the given specified protocol or configuration. The APIs can vary, from REST implementations that imply HTTP connections, to database specific ones (JDBC, ODBC, etc.) that will imply TPC connections, or even SFTP connections to a file server. This introduces the requirement for the sandbox to allow the INFINITECH component responsible for this job to access endpoints outside the sandbox using different communication protocols. Regarding the need for data access, only the data ingestion component will need to have access to those endpoints, in order to migrate the data into the central data repository. All other components in the different logical layers of the RA will retrieve the data via the data repository.

4.1.3 IoT Streaming

As we already mentioned, it becomes more crucial for modern applications in the finance and insurance sector to perform real-time analysis in order to respond to possible risks, identify opportunities or even detect fraud transactions, as they occur. This type differentiates from the others as it does not rely on data at-rest, but on streaming. As a result, content event processing over a data stream is becoming popular. Data transmitted over a stream is usually small and contains information from either a sensor deployed in a vehicle or in the soil, information coming from a finance transaction, logging information produced when a user is navigating the web or even a tweet or post on social media. We consider all these types of data as IoT, and we categorize this type for data access as IoT Streaming. The unified query processing framework of INFINITECH will require access to data streams, as it provides the means to deploy continuous queries that can perform live processing over the content of the streams, possibly making use of the other layers that allow the combination of operators targeting live data with static data at-rest. The unified query processing framework contains a streaming engine that consumes the stream from the source. The requirement for the infrastructure in this case is to allow the components that are deployed inside the sandbox to be accessible from outside of the sandbox. To concretize the scenario, the sources of the data streams must connect to the query processing framework of the platform by establishing static TPC connections to the latter. All other components in the different logical layers of the RA will retrieve data and information from this layer, so there is no need for others to be accessible from external data sources.

4.1.4 Direct access to on premise data sources

This covers scenarios where it is not enough for the integrated solution to have access only to a snapshot of the dataset that has been periodically loaded into the sandbox, but it requires access to the overall dataset. It will also cover scenarios where data cannot be migrated into the sandbox due to their volume or due to other regulatory constraints. In the case of 'big data' management, it might be better to push down the data processing to the source and grab only the results that will feed the tools and components in the analytical layer, rather than moving all the data inside the sandbox to be loaded to the central repository beforehand. We call this mode for data access as direct access to on premise sources as the sandbox needs to access directly the data source and send the processing there, and it will not migrate any data inside. The polyglot component of INFINITECH is responsible for this functionality, which lies in the processing layer of the RA, even if it pushes down to processing to the source. However, when the processing takes place, from a logical point of view, the polyglot takes care of this and all other layers in the analytical layers will make use of the latter. The requirements for the sandbox are similar with the direct data ingestion mode, but it is the polyglot component now that needs to open connections to external endpoints from the sandbox, using different communication protocols, varying from HTTP, to TPC and SFTP.

4.1.5 Blockchain data access

As the name of this access mode implies, it covers scenarios where secure access to data stored in the Blockchain is required. We are interested here in uses of Blockchain as a means of persistent storage of data, and not as a means of verifying the consensus for a given transaction. As it has been described in the corresponding deliverables of the tasks related with the Blockchain technology in WP4, the access is being granted via a specified API. Therefore, from a functional view, the components that need to access Blockchain data will have to open a connection to the provided APIs and retrieve data. The Blockchain data needs no pre-processing to the source neither can it benefit from the processing capabilities of the INFINITECH data management components. Therefore, it is the components that are lying in the analytical layer of the RA that need to be granted access to the Blockchain storage via the appropriate designated endpoint. However, from the perspective of the infrastructure that will orchestrate the automated deployment and maintenance of the integrated solution, the requirements are similar to the dynamic data ingestion: to allow components inside the sandbox to establish connections with endpoints that are located externally from the sandbox.

4.1.6 Third Party data access

In this access mode, the components and analytical tools provided by the INFINITECH platform need to retrieve data and information that has been provided by external sources. This scenario is different from the previous ones, and especially to the direct access to on premise sources, as the third party does not grant access to its entire dataset, rather than provides specific APIs that will allow others to extract only specific information. In other words, it can allow to submit a consult (i.e. how popular is the given trend in the articles of UK during the last week) and retrieve its result. The third-party organization has access to other datasets, and has already performed an initial pre-processing that generates the results of the types of consults its API allows to perform. Therefore, there is no need for the data management component that are suited in the processing layers of the RA to perform any further analysis and this information can be directly used by the analytical tools. However, from the perspective of the infrastructure that manages the sandboxes, the requirement is the same as the Blockchain: to allow the components deployed in the sandbox to open connections to APIs that are deployed outside.

4.2 Datasets management from the Blueprint reference environment perspective

Even if we have several and diverse types for data access, the requirements for the datasets and data access management tools and the techniques from the blueprint reference environment perspective (see Section 5), or rather from the testbeds and sandboxes perspective, can be described and addressed by grouping the methodologies presented in the previous paragraphs into the following macro categories:

1. **Static data ingestion**: The blueprint must be able to store static data into a persistent volume that needs to be accessible by the sandboxes. This requirement is fulfilled using the Kubernetes PV (Persistent Volume) available inside the blueprint and in particular by using the PVC (Persistent Volume Claim) that allows to split the PV according to the sandboxes needs.
2. **Dynamic data ingestion - Blockchain data access - Third party data access**: The blueprint must be able to allow the components of different layers of the logical architecture view to open connections to external endpoints. Accordingly, the components deployed inside the sandboxes need to communicate with the external endpoints. This communication is permitted by default, unless otherwise stated, through the API Gateway which forwards HTTP / HTTPS requests to and from the sandboxes.
3. **IoT Streaming - Direct access to on premise data sources**: The blueprint must be able to allow for components that have been already deployed to connect and maintain open connections with components that have been deployed inside a sandbox. Accordingly, the components that have been deployed inside a sandbox have to maintain an open connection on TPC protocol. In this case the connection are not managed by the API Gateway, but using the Kubernetes NodePort. The NodePort functionality exposes the sandbox service on each worker's IP at a static port in the range between 30000 and 32767. Each worker proxies such port (the same port number on every Node) into the sandbox service. In order to clarify this concept, let's suppose that the sandbox service works on the port 390: this port is then remapped on the port 30390 on each worker node, and in this way the sandbox can then be reached by external components and maintains an open connection on the port 30390.

5 Overview of the INFINITECH blueprint reference testbed

This chapter describes the initial design and implementation of the INFINITECH blueprint reference testbed, through the actual realization (with a full compliance) of the INFINITECH RA Development and Deployment views, in terms of the concrete specification and realization of the fundamental and target INFINITECH concepts of Testbeds, Sandboxes and Datasets management, and related tools and techniques for their effective setup and deployment in the INFINITECH pilots and validation scenarios. In other words, how the concept described in chapter 3 will be realized on the target INFINITECH infrastructure environments.

Moreover, the planned blueprint environment associated to the initial and preliminary Proof of Concept (PoC) implementation of one of the official INFINITECH pilots (at the time of writing, the WP7 **Pilot 5b: Business Financial Management (BFM) tools delivering Smart Business Advice**, owned by the partner **Bank of Cyprus (BOC)**) is also described.

5.1 Development view

In the context of WP6 and this deliverable, with respect to the Development view, it has been decided to implement DevOps (Development and Operations) processes. DevOps represents a change in IT culture, focusing on rapid IT service delivery through the adoption of agile, lean practices in the context of a system-oriented approach. DevOps emphasizes people (and culture), and seeks to improve collaboration between operations and development teams. DevOps implementations utilize technology--especially automation tools that can leverage an increasingly programmable and dynamic infrastructure from a life cycle perspective [7].

The practical implementation of DevOps goes through the CI/CD processes, which are delivered through the combined practices of Continuous Integration (CI) and Continuous Delivery (CD).

In particular:

- **Continuous Integration** is a practice where development teams frequently commit (many times per day) application code changes to a shared repository. These changes automatically trigger new builds that are then validated by automated testing (as in Development Testing, DevTest [8]), to ensure that they do not break any functionality.
- **Continuous Delivery** is an extension of the CI process. It’s the automation of the release process so that new code is deployed to target environments, typically to test environments, in a repeatable and automated fashion.

In order to fulfil the INFINITECH project goals, the CI/CD processes have been created in the context of the blueprint reference testbed environment. Moreover, to build a system working consistently as a whole, the developers writing the individual components of the INFINITECH platform need an integrated environment where they can test their components working together with the other services. To support this process, we implemented a Continuous Integration environment based on EKS (Elastic Kubernetes Service) [9], a managed Kubernetes service on the AWS public cloud. More details on EKS will be provided in the paragraph 5.2. Kubernetes is an ideal choice for a Continuous Integration environment, since it allows easy updates of deployments when new application images are built, with manifests containing deployment configurations versioned like Git [10] alongside the application source code. Furthermore, it is easy to spin up new test environments from scratch, which enables future scenarios including automated end-to-end integration testing. Build agents are also created on demand and removed when done, providing efficient resource utilization and clean environments to ensure build reproducibility.

On the target cluster, a namespace named *devops* have been created for hosting the DevOps tools, which are:

- **Gitlab** [11] is a Git repository manager that lets each developer teams collaborate on the INFINITECH's source code.
- **Jenkins** [12] is the de-facto standard open source automation server for orchestrating CI/CD workflows.
- **Sonatype Nexus** [13] is a popular artefact repository that also works as a Docker registry, as required in our case.
- **OpenLDAP** [14] is used as the single user directory for all tools, centralizing authentication and simplifying management of developer accounts.
- **Helm** [15] is a package manager that streamlines installing and managing Kubernetes applications.

Figure 12 shows how CI/CD works for a specific partner (e.g. Partner "A"). When a developer pushes new component code, Gitlab invokes a webhook on Jenkins, which starts any job affected by the code changes. The job builds the component, runs unit tests and, if everything has worked in a proper way, builds an updated Docker image and pushes it to Nexus. The following step is deploying the updated component in the specific partner namespace; in fact, we will have as many namespaces as the partners in order to maintain the correct isolation between all INFINITECH partners. In order to deploy the component Helm manager will be used. At the end of the process, Jenkins sends a notification to a dedicated CI/CD channel on the INFINITECH Slack [16] project, so that developers are informed that a new build occurred and whether it was successful or not. In case of errors, developers will have to inspect the build logs, find the problem and correct it. In case of success, developers will go ahead and test that the new version works correctly in the test environment.

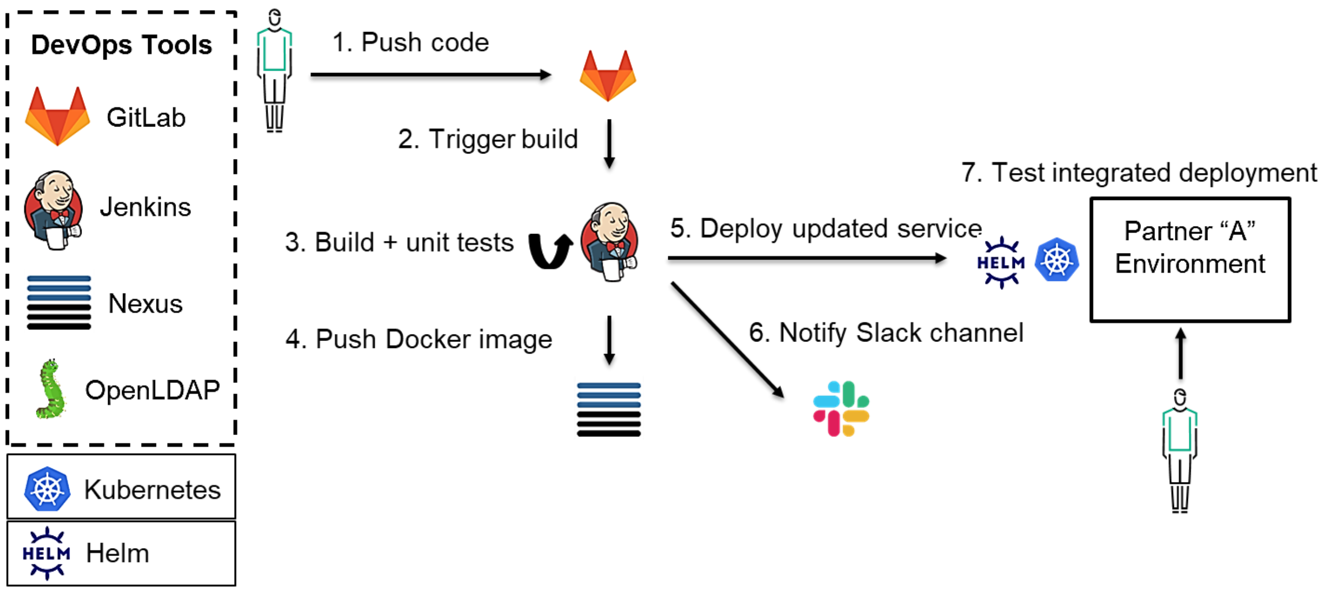

Figure 12 CI/CD workflow

Moreover, in the following iterations of WP6 T6.2 and T6.3 tasks and activities (e.g. in the future deliverables D6.5 and D6.6), we intend to enhance the process by adopting a DevSecOps [17] approach and including the related tools like **Sonarqube** [18]in the CI/CD pipeline.

DevSecOps aims to include security in the software development life cycle from the beginning, following the same principles of DevOps. Security is then considered throughout the process and not just as an afterthought at the end of it, so that different kinds of security checks are executed continuously and automatically, giving developers quick feedback if the latest changes introduced a vulnerability that must be corrected.

DevSecOps of course requires that security experts work side by side with developers and operations to make sure that security requirements are addressed and best practices followed, in addition to validating product design and architecture.

Moreover, in this context, in order to facilitate the Machine Learning (ML) development and testing, we have also planned to introduce (again, in the following iterations of WP6 T6.2 and T6.3 tasks and activities), the MLOps [19](a compound of Machine Learning and IT Operations) methodology focused on:

- Facilitate communication and collaboration between teams.
- Improve model tracking, versioning, monitoring and management.
- Standardize the machine learning process to prepare for increasing regulation and policy.

This add-on enhanced methodology will enable us to automate the porting of the machine learning algorithms, as much as possible, in production environments.

Putting this into practice is often very complicated, because ML processes are often based on heterogeneous environments.

Therefore, the first step towards MLOps requires the standardization of these environments as much as possible, and in this respect the Kubernetes technology and containers provide the abstraction, scalability, portability, and reproducibility required to run the same piece of software in all these environments. As a second step, it is necessary to make standard the workflows used for the construction and building of the ML models.

In this sense, we have selected the **Kubeflow** [20] platform as candidate for integration in our blueprint reference testbed, in order to provide an infrastructure to build models and capable of enabling the portability of these models and workflows. In particular, ML workflows are defined as Kubeflow pipelines and a pipeline consists of these steps:

- Data preparation.
- Training.
- Testing.
- Serving.

Each step is a container and the output of each step is the input of the following step. Once compiled, this pipeline is portable across different environments.

5.2 Deployment view

In the context of WP6 and this deliverable, with respect to the Deployment view, it has been decided to create the INFINITECH blueprint reference testbed on the AWS (Amazon Web Services) public provider. In particular, such blueprint will be hosted on the Amazon EKS (Elastic Kubernetes Service). Amazon EKS is a managed service that allows to use a Kubernetes environment without the need to install, operate, and maintain Kubernetes control plane or nodes.

The EKS control plane (Figure 13) is composed by at least two API server nodes and three etcd nodes that run across three Availability Zones within a Region. In case of issue Amazon EKS automatically detects and replaces unhealthy control plane instances, restarting them across the Availability Zones within the Region as needed. From the security perspective all information stored on etcd nodes and associated Amazon EBS (Elastic Block Store) volumes is encrypted using AWS KMS (Key Management Service).

Regarding the data plane, we will start using 2 nodes, but we will also define the Auto Scaling rules in order to scale the Kubernetes cluster according of the real usage of the platform.

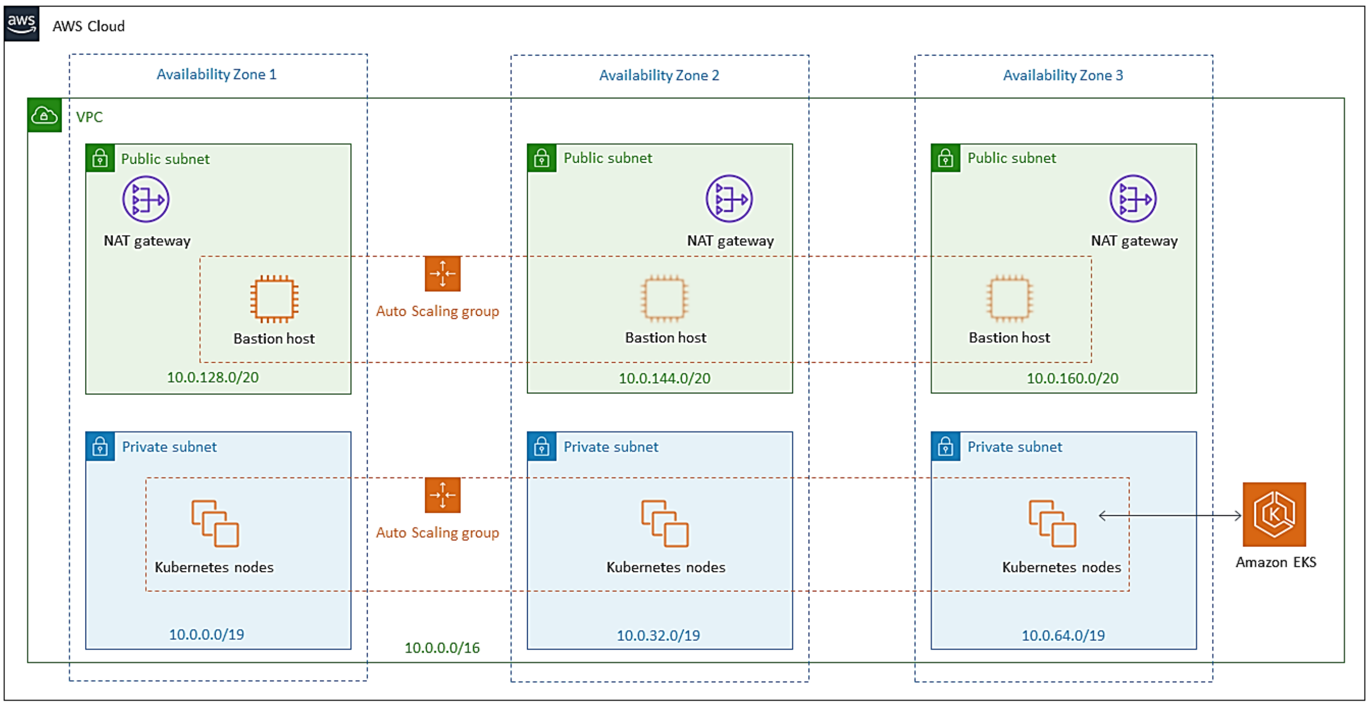

Figure 13 EKS Control Plane deployment[3]

5.2.1 Creation of the EKS INFINITECH cluster

The creation of EKS INFINITECH cluster involves four principal steps:

- Create a specific IAM (Identity and Access Management) Role able to create a provisioning EKS cluster.
- Create the VPC (Virtual Private Cloud).
- Create the EKS Control plane.
- Create the Worker node.

To perform all the steps we will leverage the AWS console and where possible also the CloudFormation templates that allow us to replicate the installation and configuration anytime in very straightforward way. For the creation of the IAM role we will use the following Amazon CloudFormation template:

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Amazon EKS Cluster Role'


Resources:

  eksClusterRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
```

[3] Picture from https://aws.amazon.com/it/quickstart/architecture/amazon-eks/

```
          Principal:
            Service:
            - eks.amazonaws.com
          Action:
          - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy

Outputs:

  RoleArn:
    Description: The role that Amazon EKS will use to create AWS resources for
Kubernetes clusters
    Value: !GetAtt eksClusterRole.Arn
    Export:
      Name: !Sub "${AWS::StackName}-RoleArn"
```

To create the VPC for EKS we will use the following template:

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Amazon EKS Sample VPC - Private and Public subnets'

Parameters:

  VpcBlock:
    Type: String
    Default: 192.168.0.0/16
    Description: The CIDR range for the VPC. This should be a valid private (RFC
1918) CIDR range.

  PublicSubnet01Block:
    Type: String
    Default: 192.168.0.0/18
    Description: CidrBlock for public subnet 01 within the VPC

  PublicSubnet02Block:
    Type: String
    Default: 192.168.64.0/18
    Description: CidrBlock for public subnet 02 within the VPC

  PrivateSubnet01Block:
    Type: String
    Default: 192.168.128.0/18
    Description: CidrBlock for private subnet 01 within the VPC

  PrivateSubnet02Block:
    Type: String
    Default: 192.168.192.0/18
    Description: CidrBlock for private subnet 02 within the VPC

Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      -
        Label:
          default: "Worker Network Configuration"
        Parameters:
          - VpcBlock
          - PublicSubnet01Block
          - PublicSubnet02Block
          - PrivateSubnet01Block
          - PrivateSubnet02Block

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock:  !Ref VpcBlock
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-VPC'

  InternetGateway:
    Type: "AWS::EC2::InternetGateway"

  VPCGatewayAttachment:
    Type: "AWS::EC2::VPCGatewayAttachment"
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
      - Key: Name
        Value: Public Subnets
      - Key: Network
        Value: Public

  PrivateRouteTable01:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
      - Key: Name
        Value: Private Subnet AZ1
      - Key: Network
        Value: Private01

  PrivateRouteTable02:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC
      Tags:
      - Key: Name
        Value: Private Subnet AZ2
      - Key: Network
        Value: Private02

  PublicRoute:
    DependsOn: VPCGatewayAttachment
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  PrivateRoute01:
    DependsOn:
    - VPCGatewayAttachment
    - NatGateway01
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable01
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway01

  PrivateRoute02:
    DependsOn:
    - VPCGatewayAttachment
    - NatGateway02
    Type: AWS::EC2::Route
    Properties:
      RouteTableId: !Ref PrivateRouteTable02
      DestinationCidrBlock: 0.0.0.0/0
      NatGatewayId: !Ref NatGateway02

  NatGateway01:
    DependsOn:
    - NatGatewayEIP1
    - PublicSubnet01
    - VPCGatewayAttachment
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt 'NatGatewayEIP1.AllocationId'
      SubnetId: !Ref PublicSubnet01
      Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-NatGatewayAZ1'

  NatGateway02:
    DependsOn:
    - NatGatewayEIP2
    - PublicSubnet02
    - VPCGatewayAttachment
    Type: AWS::EC2::NatGateway
    Properties:
      AllocationId: !GetAtt 'NatGatewayEIP2.AllocationId'
      SubnetId: !Ref PublicSubnet02
      Tags:
      - Key: Name
        Value: !Sub '${AWS::StackName}-NatGatewayAZ2'

  NatGatewayEIP1:
    DependsOn:
    - VPCGatewayAttachment
    Type: 'AWS::EC2::EIP'
    Properties:
      Domain: vpc

  NatGatewayEIP2:
    DependsOn:
    - VPCGatewayAttachment
    Type: 'AWS::EC2::EIP'
    Properties:
      Domain: vpc

  PublicSubnet01:
    Type: AWS::EC2::Subnet
    Metadata:
      Comment: Subnet 01
    Properties:
      MapPublicIpOnLaunch: true
      AvailabilityZone:
        Fn::Select:
        - '0'
        - Fn::GetAZs:
            Ref: AWS::Region
      CidrBlock:
        Ref: PublicSubnet01Block
      VpcId:
        Ref: VPC
      Tags:
      - Key: Name
        Value: !Sub "${AWS::StackName}-PublicSubnet01"
      - Key: kubernetes.io/role/elb
        Value: 1

  PublicSubnet02:
    Type: AWS::EC2::Subnet
    Metadata:
      Comment: Subnet 02
    Properties:
      MapPublicIpOnLaunch: true
      AvailabilityZone:
        Fn::Select:
        - '1'
        - Fn::GetAZs:
            Ref: AWS::Region
      CidrBlock:
        Ref: PublicSubnet02Block
      VpcId:
        Ref: VPC
      Tags:
      - Key: Name
        Value: !Sub "${AWS::StackName}-PublicSubnet02"
      - Key: kubernetes.io/role/elb
        Value: 1

  PrivateSubnet01:
    Type: AWS::EC2::Subnet
    Metadata:
      Comment: Subnet 03
    Properties:
      AvailabilityZone:
        Fn::Select:
        - '0'
        - Fn::GetAZs:
            Ref: AWS::Region
      CidrBlock:
        Ref: PrivateSubnet01Block
      VpcId:
        Ref: VPC
      Tags:
      - Key: Name
        Value: !Sub "${AWS::StackName}-PrivateSubnet01"
      - Key: kubernetes.io/role/internal-elb
        Value: 1

  PrivateSubnet02:
    Type: AWS::EC2::Subnet
    Metadata:
      Comment: Private Subnet 02
    Properties:
      AvailabilityZone:
        Fn::Select:
        - '1'
        - Fn::GetAZs:
            Ref: AWS::Region
      CidrBlock:
        Ref: PrivateSubnet02Block
      VpcId:
        Ref: VPC
      Tags:
      - Key: Name
        Value: !Sub "${AWS::StackName}-PrivateSubnet02"
      - Key: kubernetes.io/role/internal-elb
        Value: 1

  PublicSubnet01RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
```

```
    Properties:
      SubnetId: !Ref PublicSubnet01
      RouteTableId: !Ref PublicRouteTable

  PublicSubnet02RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnet02
      RouteTableId: !Ref PublicRouteTable

  PrivateSubnet01RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet01
      RouteTableId: !Ref PrivateRouteTable01

  PrivateSubnet02RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet02
      RouteTableId: !Ref PrivateRouteTable02

  ControlPlaneSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Cluster communication with worker nodes
      VpcId: !Ref VPC

Outputs:

  SubnetIds:
    Description: Subnets IDs in the VPC
    Value: !Join [ ",", [ !Ref PublicSubnet01, !Ref PublicSubnet02, !Ref
PrivateSubnet01, !Ref PrivateSubnet02 ] ]

  SecurityGroups: eith
    Description: Security group for the cluster control plane communication with
worker nodes
    Value: !Join [ ",", [ !Ref ControlPlaneSecurityGroup ] ]

  VpcId:
    Description: The VPC Id
    Value: !Ref VPC
```

The previous template creates the VPC with two public and two private subnets. One public and one private subnets are deployed to the same Availability Zone. The second public and private subnets are deployed to a second Availability Zone in the same Region. We have chosen this solution because it is that recommended by AWS for critical environment.

After that the VPC is available, it’s possible to proceed with the EKS Control plane creation that can be done directly from the AWS console at https://console.aws.amazon.com/eks/home#/clusters through the input of the following parameters:

- Name: **INFINITECH-BP**
- Kubernetes version: **1.17**
- Public and private: **true**
- role: **eksClusterRole**(create with in the previous step)
- Encryption: false(Activate if is necessary)
- Subnet: **all**
- Security groupsInfo: **ControlPlane**
- Access point Public: **True**

When the EKS Control plane is available (the creation takes about 15 minutes), it is possible to create the worker nodes.

In order to add the nodes, it is mandatory to create a specific IAM role named *WorkerRole* with the following permissions:

- AmazonEKSWorkerNodePolicy.
- AmazonEKS_CNI_Policy.
- AmazonEC2ContainerRegistryReadOnly.

Moreover, to facilitate the role management we will add the tag value *INFINITECHBPNode*.

At this time, it is possible to create from the same AWS console the worker nodes by clicking on the Compute tab and by entering these values:

- Name: **T3xlarge_16GB4-4vCPU**
- Node IAM role name: **WorkerRole**
- AMI type: **Amazon Linux 2 (AL2_x86_64)**
- Instance Type: **t3a.xlarge(4vCPU/16GB)**
- Disk size: **50 GB**
- Minimum size: **2**
- Maximum size: **3**
- Desired size: **2**
- Subnets: **private-sub01; private-sub02**
- Allow remote access to nodes: **true**
- Allow remote access from: **all**

At this stage, the Kubernetes cluster is ready to use.

5.2.2 Namespace, Network and Quote policies

As previously described in chapter 3.5, we have decided to implement the INFINITECH Sandbox concept leveraging the Kubernetes namespace feature. Therefore, the first step is to create a dedicated Namespace for each of the INFINITECH target pilots that can be leveraged by each of the owning partners to develop and test their own pilot applications.

To create the namespace we have defined a Kubernetes YAML template like the following:

```
kind: Namespace
apiVersion: v1
metadata:
  name: Pilot1
  labels:
    name: Pilot1
```

The first namespace that has to be created has been named *devops*, aimed to host all the DevOps tools.

Some of these tools (like Jenkins and Gitlab) need to be exposed on the Internet for convenient access during the development and testing phases. All the endpoints will be HTTPS based, with free certificates generated by Let's Encrypt [21], so we do not need to set up a Certification Authority or buy certificates from commercial CAs.

However, we do need public DNS entries on the project domain, mapped to a public IP that exposes our services outside of the Kubernetes cluster.

In order to get the public IP, we will use the AWS ELB (Elastic Load Balancer). This is provisioned automatically when creating a Kubernetes Service with type LoadBalancer. In our case, this happens while configuring the NGINX Ingress Controller. The NGINX Ingress Controller that will be deployed will have a dedicated namespace named *ingress-nginx.*

To implement the network policies to isolate each namespace from each other, and eventually also to manage the connection between different PODs within the same namespace (i.e. to guarantee the security requirements), we will implement the Cilium as CNI (Container Network Interface) on EKS.

In order to do that, we will remove the AWS CNI and install the Cilium following these steps:

```
kubectl -n kube-system delete daemonset aws-node
helm repo add cilium https://helm.cilium.io/

helm install cilium cilium/cilium --version 1.7.9 \
  --namespace kube-system \
  --set global.eni=true \
  --set global.egressMasqueradeInterfaces=eth0 \
  --set global.tunnel=disabled \
  --set global.nodeinit.enabled=true
```

After the previous step, it is possible to isolate the namespace implement the following rule (the rules is written in the YAML format):

```
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "isolate-pilot1"
  namespace: pilot1
spec:
  endpointSelector:
    matchLabels:
      {}
  ingress:
  - fromEndpoints:
    - matchLabels:
        {}
```

Moreover, we will apply the resource quote on memory and CPU that each namespace can consume with this YAML template:

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-namespace
spec:
```

```
hard:
  requests.cpu: "800m"
  requests.memory: 2Gi
  limits.cpu: "1"
  limits.memory: 3Gi
```

After the execution of the previous steps, the Deployment view has been completed, as shown in Figure 14.

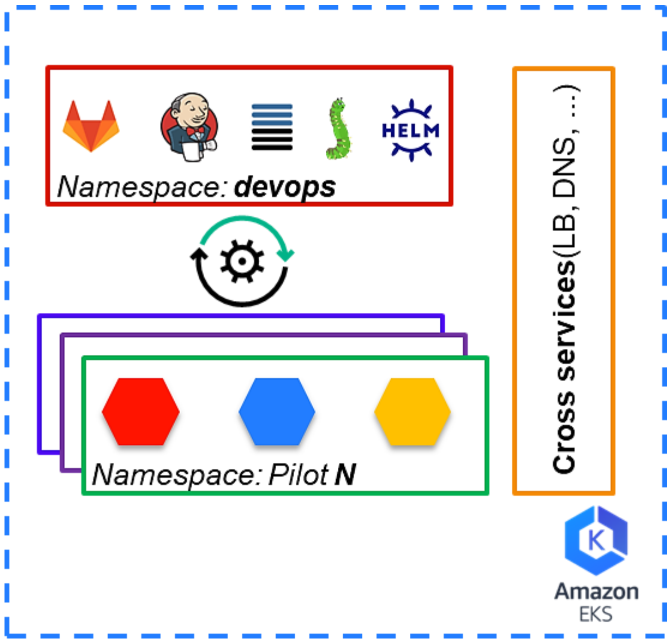

Figure 14 INFINITECH blueprint reference testbed

5.2.3 How to recreate the blueprint testbed for a specific INFINITECH Pilot

One of the most powerful capabilities and features that are enabled by the technological choices we have made and that are described in the previous sections, is that the blueprint reference testbed that we have created on AWS can be recreated from scratch, with respect to the Deployment view perspective, by each of the partners for their own pilots in two possible ways (see Figure 15):

I. On the same AWS cloud provider (a concrete realization is described in section 5.3).
II. In a bare metal environment, leveraging their on-premise private data centre infrastructure or the shared NOVA's Data Centre.

The two ways are a little bit different each other, because:

I. In the first case, the recreation of the cluster can be done in fully automated way using the `eksctl` tool provided by AWS.
II. In the second case. it is possible to recreate the cluster in a partially automated way, because in this case as mandatory prerequisite it is necessary to manually create the entire infrastructure that will host the Kubernetes cluster, and only afterwards it is possible to create the cluster using automated tools like `Kubespray` provided by Kubernetes itself.

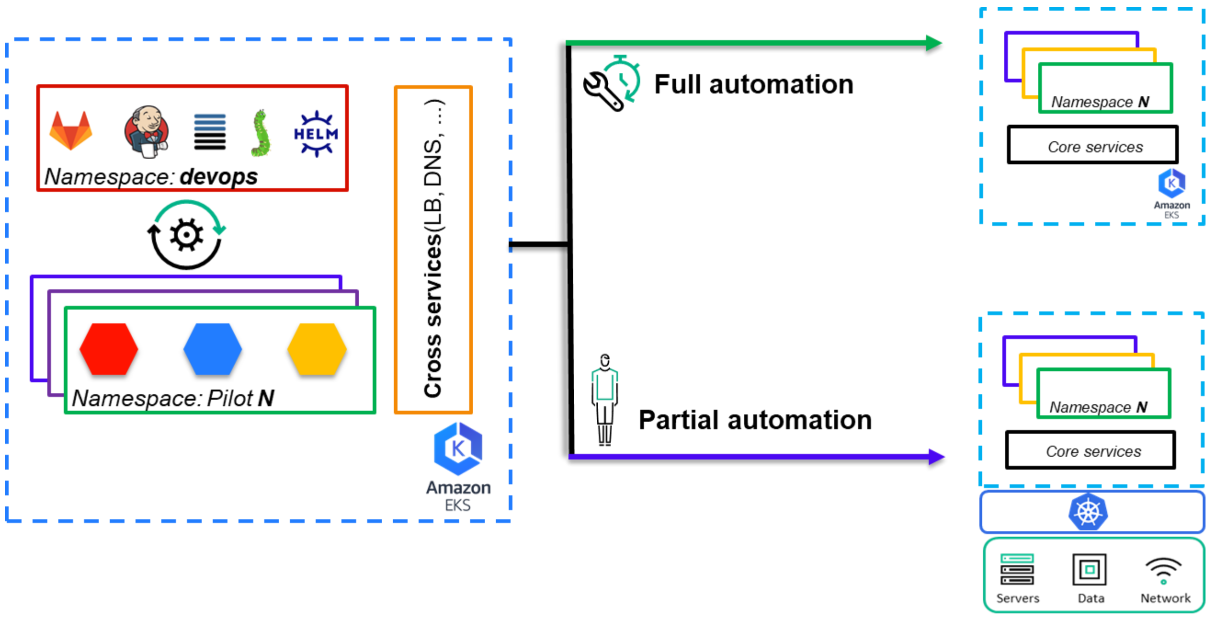

Figure 15 Blueprint environment recreation ways

Nevertheless, one of the major objectives of having provided a blueprint reference testbed is exactly the powerful concept explained above: a potential easy and straightforward replication of it for all the INFINITECH target pilots' environments.

5.3 Blueprint environment for Pilot 5b: Bank of Cyprus

As mentioned at the begin of chapter 4.2, as a concrete INFINITECH blueprint environment associated to one of the official INFINITECH pilots, the consortium has selected (at the time of writing) the WP7 **Pilot 5b: Business Financial Management (BFM) tools delivering a Smart Business Advise**, owned by the partner **Bank of Cyprus (BOC)**, and in particular its initial and preliminary Proof of Concept (PoC) implementation.

This section describes the preliminary concrete realization of such PoC and also setup the basis for the future applications of the concepts described in the previous section 5.2.3 for all the INFINITECH target pilots' environments.

5.3.1 Pilot Objectives

Most of today's Financial Management tools for Small Medium Enterprises (SMEs) are geared towards analysing only past transactions, making such tools inadequate in today's world. Today, SMEs and their customers alike, demand just-in-time processing, transparency and personalized services to assist SME owners not only in understanding better their SME business/financial health but also to be able to decide on the next best action to take.

Thus, Pilot 5b aims to assist SME clients of Bank of Cyprus in managing their financial health in the areas of cash flow management, continuous spending/cost analysis, budgeting, revenue review and VAT provisioning, all by providing a set of AI-powered Business Financial Management tools and harnessing available data to generate personalized business insights and recommendations. Machine learning algorithms, predictive analytics and AI-based interfaces will be utilized to develop a kind of smart virtual advisor with the aim to minimize SME business analysis effort, to focus on growth opportunities and to optimize cash flows performance.

5.3.2 Pilot Workflow

The pilot workflow can be analysed starting from the datasets that it has to manage. Some of the available datasets require real time data collection, while in others historical data collection is sufficient to provide actionable business insights. In detail, transaction and account data related to the respective SME will be drawn from BOC's repository by a real time/historical data collector as well as transaction and account data from Open Banking (PSD2), as well as BOC's customer data, will utilize a historical data collector. In addition, an external data collector will also be used in order to integrate other related Open Banking/macroeconomic data. The SMEs data source (e.g. ERP/Accounting system) utilization remains optional as consent is required for the collection and processing of such data and its cloud availability being required. Accordingly, it is possible summarize the involved data sources involved in the following list:

- Transaction Data from Open Banking (PSD2).
- Transaction Data from SMEs (optional).
- Other Data (Market).
- Other Data from SMEs (optional).
- Accounts Data from BOC.
- Accounts Data from Open Banking.
- Customer Data from BOC.
- Direct Input from SMEs.

In the target pilot PoC, all data except external macroeconomic data will be pseudonymised (by tokenization) before being uploaded to the IRA.

The cloud Data Repository (within IRA) will then store all collected data (along with the generated insights), past SME financial actions (to measure at what degree the SME actions reflect the recommended insights), as well as minimum user input that is required. A continuous data streaming will connect the Data Repository with the various deployed BFM tools (machine learning algorithms), which would allow the retraining of the respective AI models and the generation of useful insights and recommended actions. A reverse data pseudonymisation will then be applied before the processed data move to the bank middleware component that contains composite APIs and produces push notifications, all of which will be offered to the SMEs via Android, iOS and web applications. Upon SME user login, the IRA is also accessed, insights/recommendations picked up from the cloud data repository and provided to the SME user.

The pilot's workflow is depicted in the below Figure 16.

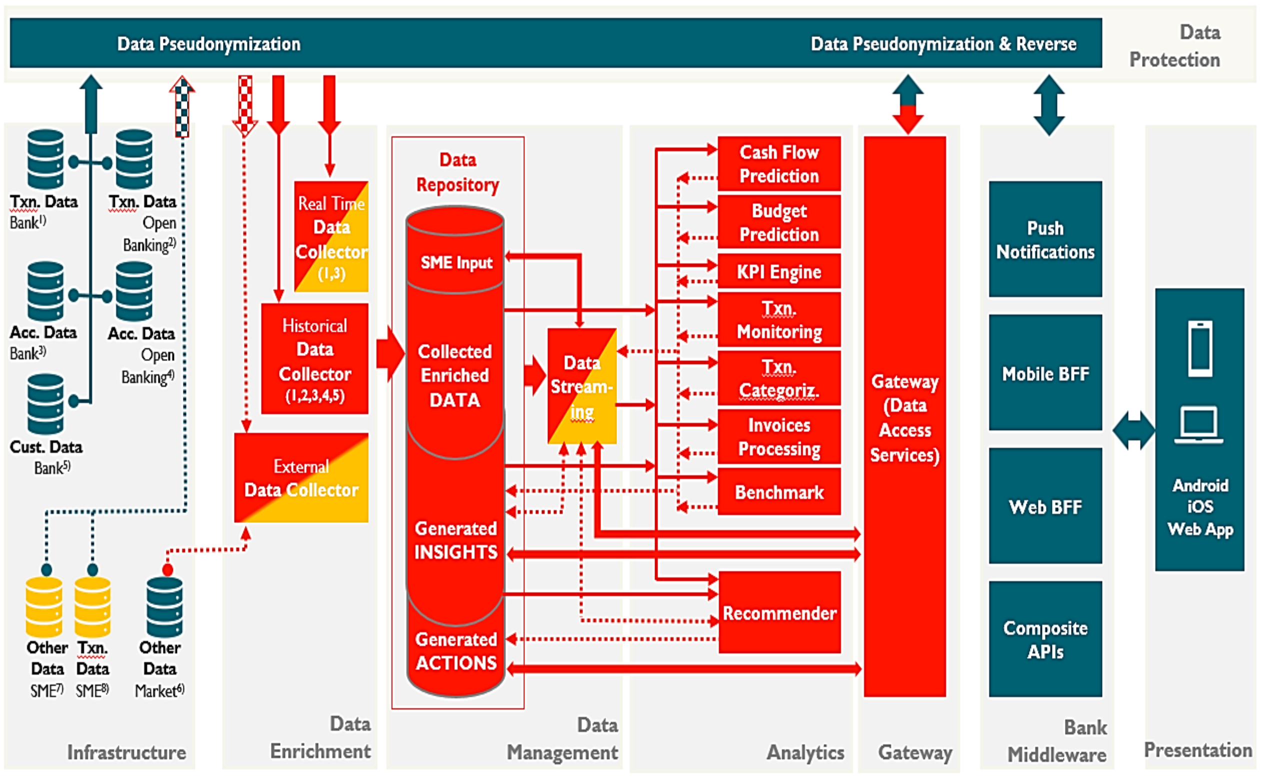

Figure 16 Pilot 5B workflow

5.3.3 Blueprint reference testbed implementation

As stated before, the blueprint reference testbed (Blueprint_v1) refers to the INFINITECH BOC pilot and its initial and preliminary PoC version (Prototype_v1) aims to accommodate the first version of the pilot. Accordingly, the PoC will not use all the available components of the INFINITECH platform, but just a selected subset.

In general, the blueprint testbed basic assumptions are:

- The pilot does not utilize Blockchain technology, so all the Blockchain-oriented components will not be integrated.
- Data migrated to the blueprint testbed is owned by BOC and will be pseudonymised (using the tokenization technique) before entering the INFINITECH ecosystem. There is no need for an INFINITECH anonymizer for the Prototype_v1 PoC.
- The first PoC version of the pilot (Prototype_v1) may not fully exploit the first version of the pilot Blueprint (Blueprint_v1). There might be additional INFINITECH components (especially regarding the Data Management layer) that remain to be decided.

Prototype_v1 and Blueprint_v1 versions will include the components listed in Table 2.

Table 2 Prototype_v1 vs Blueprint_v1 components

Group	Prototype_v1	Blueprint_v1
On Premise Data source Layer and Data Management	(to be decided)	(to be decided)
Analytics Layer	Cash Flow Prediction ,	Cash Flow Prediction ,

	Transaction Categorization	Transaction Categorization
Data Models and Semantics	Preprocessing	Preprocessing
Data Security and Privacy	-	Data Anonymization
Interfaces	Proxy , API	Proxy , API

The next version of the PoC prototype (Prototype_v2) will include stream processing, possibly anonymization and some complementary components of the Analytics group.

The ML/DL models of the Analytics Components (Cash Flow Prediction, Transaction Categorization) will be trained offline and then afterwards loaded to the blueprint analytics component. The models' training process (including the validation and evaluation processes) is depicted in Figure 17.

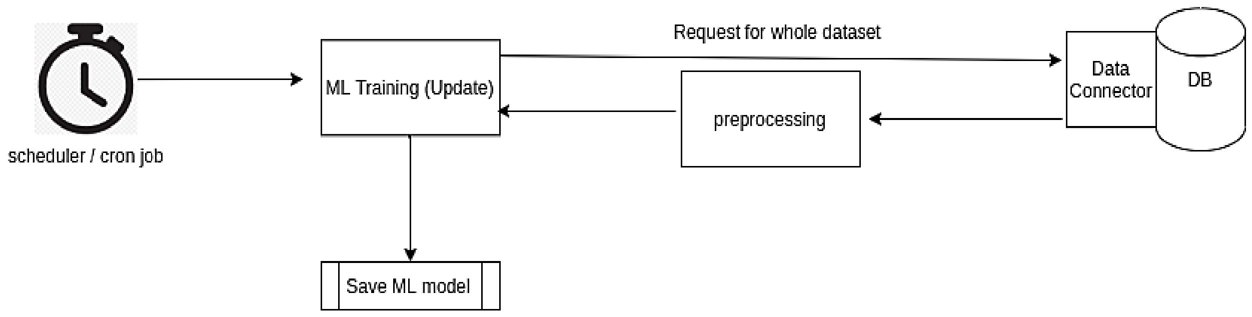

Figure 17 Models training process

Given the aforementioned points, a REST API will be developed on top, which will deliver real-time information when invoked. This data will be forwarded to the specific analytics component (Cash Flow Prediction, Transaction Categorization) via a REST API/TCP connection or other message broker.

In order to run the pilot inside the blueprint environment a dedicated sandbox named **pilot5b** has been created. The components deployed in such sandbox are:

- Analytics.
- Pre-processing.
- Anonymizer.

The client connections from the external world towards such a sandbox and all sandboxes deployed in the blueprint are managed through an API Gateway (GW) (see Figure 2), which is a key component of the IRA Interface layer. The API GW in the blueprint environment is based on the Istio [22] software and is deployed in a dedicated sandbox named **istio-system**.

The process workflow implies that when a user sends a (HTTPS) request it is forwarded by the API GW to the Analytics POD, which will then request the appropriate historical data, stored on premise, from the "Data Management" layer via a data connection/REST API or other message broker. The retrieved data will be first anonymized and then pre-processed based on the approach used for the training procedure. The data will then be forwarded back to the Analytics component in order to be injected in the models to infer the outcome of each model. Finally, the results will be either returned to the user or saved to the internal Data Base, available within the Analytics POD, according to the scope of the process.

The proposed blueprint architecture is illustrated in Figure 18.

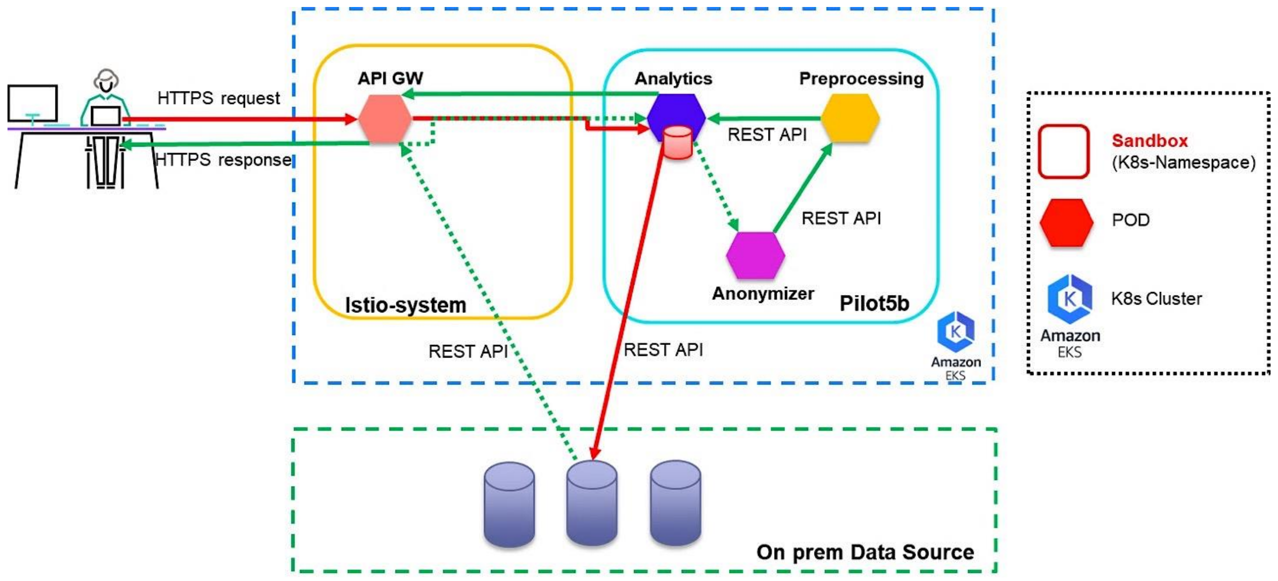

Figure 18 Pilot 5b blueprint reference architecture

6 Conclusions

This document has reported the results of INFINITECH WP6 Tasks T6.2 “Mechanisms and Tools for Tailored Sandboxes Provision and Configuration” and T6.3 “Integrated Management of Testbeds' Datasets”, achieved during the first phase of the project, i.e. the initial specification of tools and techniques that will be leveraged to implement the testbeds and sandboxes concepts and the management of datasets within the INFINITECH project.

The document is the accompanying textual specification of the other major deliverable result: the initial release of the proposed tools and techniques integrated and deployed into the INFINITECH blueprint reference testbed setup, designed and implemented upon one of the target INFINITECH infrastructures. The document and the developed INFINITECH blueprint reference testbed setup constitutes the overall deliverable output.

The achieved results provide key contributions for the fulfilment of the 1st major WP6 milestone (MS9 – First Version of ALL Sandboxes & Testbeds Available – foreseen for M16 of the project).

The work has been carried out in close cooperation and coordination with the other INFINITECH WP6 tasks and work packages 2-3-4-5 tasks and partners, taking into account and integrating the delivered results and concepts (e.g. the INFINITECH Reference Architecture proposed by WP2) in a coherent and uniform manner.

The overall progress of such WP6 tasks will be one of the major drivers of the INFINITECH work package dedicated to the Large Pilots Operations and Stakeholders Evaluation of the proposed Financial and Insurance Services (WP7).

The WP6 work on Tasks 6.2 and 6.3 will continue without any interruption in the next period of the project, evolving and enriching the proposed tools and techniques with additional capabilities and features that will take into account the latest evolutions of relevant technologies occurring during the related project timeframe. Such work will lead to other two major releases of the deliverable, which will be fully documented respectively in deliverable D6.5 (Tools and Techniques for Tailored Sandboxes and Management of Datasets-II), planned for M22, and deliverable D6.6 (Tools and Techniques for Tailored Sandboxes and Management of Datasets-III), planned for M30.

Finally, with respect to the specific objectives and KPIs set for WP6 and related tasks in the project DoA, the Table 3 summarize how the work done in this deliverable addresses them.

Table 3 – Mapping of DoA/Tasks Objectives with Deliverable achievements

Objectives	**Comment**
OBJECTIVE 6 – TESTBEDS AND TAILORED SANDBOXES FOR BIGDATA & IOT EXPERIMENTATION, TESTING, INNOVATION AND STANDARDIZATION IN FINANCE & INSURANCE	The approach, mechanisms and tools described in D6.4 enables the definition and deployment of the testbeds and sandboxes envisioned by the objective.

Table 4 – Mapping of DoA/Tasks KPIs with Deliverable achievements

KPI	Comment
KPI 1 Testbeds to be Established >= 9 (>=8 in Incumbent organizations and >=1 (EU-wide) testbed for FinTech/InsuranceTech firms); **KPI 2 Distinct datasets (assets) to be shared through the testbeds >=25;** **KPI 3 Tailored Sandboxes to be developed & customized based on the project's tools>=14 (i.e. equal to the number of pilots).**	The approach, mechanisms and tools described in D6.4 incorporates by design the proper scalability features that enable the definition, deployment and provisioning of the target numbers of testbeds, sandboxes and datasets envisioned by the KPIs.

7 Appendix A: Literature

[1] K. Philippe, "“Architectural Blueprints - The “4+1” View Model of Software Architecture”," *IEEE Software,* vol. 12, pp. pp. 42-50, November 1995, pp. 42-50.

[2] R. M. M. M. M. A. Irakli Nadareishvili, Microservice Architecture: Aligning Principles, Practices, and Culture, O'Reilly Media, Inc., 2016.

[3] Google, “Containers,” Google, 2018. [Online]. Available: https://cloud.google.com/containers.

[4] Kubernetes, “What is kubernetes?,” Kubernetes, 5 August 2020. [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.

[5] Docker, “What container,” 2020. [Online]. Available: https://www.docker.com/resources/what-container.

[6] AWS, “What is aws,” Amazon, 2020. [Online]. Available: https://aws.amazon.com/it/what-is-aws/.

[7] Gartner, “DevOps Gartener Glossary,” Gartner, 2020. [Online]. Available: https://www.gartner.com/en/information-technology/glossary/devops.

[8] “DevTest and DevOps for microservices,” Microsoft, 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/dev-test-microservice.

[9] AWS, “Amazon EKS,” Amazon, 2020. [Online]. Available: https://aws.amazon.com/eks/?nc1=h_ls.

[10] Git, “Git,” Software Freedom Conservancy, 2020. [Online]. Available: https://git-scm.com/about.

[11] GitLab, “Gitlab docs,” GitLab, 2020. [Online]. Available: https://docs.gitlab.com/charts/installation/deployment.html.

[12] Jenkins, “Jenkins,” [Online]. Available: https://jenkins.io/.

[13] S. Nexus, “Sonatype Nexus OSS,” [Online]. Available: https://www.sonatype.com/nexus-repository-oss.

[14] OpenLDAP, “OpenLDAP,” [Online]. Available: https://www.openldap.org/.

[15] HELM, “Quickstart,” HELM, 2020. [Online]. Available: https://helm.sh/docs/intro/quickstart/.

[16] Slack, “Slack features,” Microsoft, [Online]. Available: https://slack.com/intl/en-it/features.

[17] D. G. Neil MacDonald, “12 things to get right for successful devsecops,” Gartner, December 2019. [Online]. Available: https://www.gartner.com/en/documents/3978490/12-things-to-get-right-for-successful-devsecops.

[18] Sonarqube, “Sonarqube architecture,” SonarSource SA, 2020. [Online]. Available: https://docs.sonarqube.org/latest/architecture/architecture-integration/.

[19] Google, “MLOps: Continuous delivery and automation pipelines in machine learning,” Google, 2019. [Online]. Available: https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning.

[20] Kubeflow, “About Kubeflow,” Kubeflow, 2020. [Online]. Available: https://www.kubeflow.org/docs/about/kubeflow/.

[21] L. Encrypt, “Let’s Encrypt,” [Online]. Available: https://letsencrypt.org/.

[22] Istio, “What is Istio,” Istio, 2020. [Online]. Available: https://istio.io/latest/docs/concepts/what-is-istio/
.