EUSEC

EU SECURITY CERTIFICATION

EUROPEAN SECURITY CERTIFICATION FRAMEWORK

PILOT PREPARATION

PROJECT NUMBER: 731845

PROJECT TITLE: EU-SEC

AUTHOR: MARIO MAAWAD, RAUL GRACIA, RAMON MARTIN DE POZUELO

PARTNERS CONTRIBUTED: CAIXA, CSA, FRAUNHOFER, SIXSQ, NIXU, FABASOFT

This project has received funding from the European Union's HORIZON Framework Program for research, technological development and demonstration under grant agreement no 731845

EXECUTIVE SUMMARY

This document is part of WP5 in the EU-SEC project.

First, we present a proposal for the continuous auditing based certification pilot in the financial sector (CaixaBank). In particular, the use-case that motivates this "Financial Pilot" of the EU-SEC project is "financial information sharing" (FISH). That is, the management and exchange of sensitive documents among financial institutions (e.g., banks, insurance companies) and regulatory authorities (e.g., Central European Bank), which is becoming increasingly relevant in the recent years. The objective of this pilot is to allow us to perform continuous auditing of a financial information sharing application in the Cloud to simplify life to involved parties, while having guarantees that the Cloud provider continuously meets with the requirements to run such a service.

Based on this use case, detailed information is provided on the pilot design and the roles played by the EU-SEC framework components presented in WP3. Moreover, the interactions with all the EU-SEC framework components in order to meet the requirements of this use-case are described. For instance, this includes the translation of requirements into automated controls in Clouditor, the involvement of Starwatch, and the secure storage of evidences, among other issues.

ABBREVIATIONS

Table 1-1 Abbreviations

Abbreviation	Description
AWS	Amazon Web Services
CCM	Cloud Control Matrix
CIMI	Cloud Infrastructure Management Interface
CSP	Cloud Service Provider
FISH	Financial Information Sharing
GDPR	General Data Protection Regulation
ISO	International Organization for Standardization
LOPD	"Ley Orgánica de Protección de Datos" (Spanish)
PCI DSS	Payment Card Industry Data Security Standard

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

In the last decade, we have witnessed a strong centralization wave of data and computing services where **Cloud providers** have exhibited a leading role. That is, services such as Amazon Web Services (AWS), Microsoft Azure and IBM Cloud provide a virtually infinite pool of IT resources to customers, which can get rid of important up-front investments in computing infrastructure (i.e., "pay-as-you-go"), as well as the complexity and costs of its associated maintenance. Such a paradigm change has leveraged very attractive opportunities for companies and organizations of all kind.

But not only this: **security** is becoming a cornerstone service for large Cloud providers. In fact, "cloud security" has been (and still is) a hot topic for both the research community and the industry in the last years. As a consequence of these efforts, today, an increasingly larger fraction of customers fell that major Cloud providers offer reliable and secure IT infrastructures to run production services. This favors the adoption of Cloud services by customers.

However, there are still companies from specific sectors that are **reluctant to move their core services to the Cloud**, such as health companies and financial institutions, to name a few. In this sense, a main roadblock for these companies is not security (i.e., public Cloud may implement even better security mechanisms than a customer's in-house facilities), but **regulatory compliance**. The roots of this problem lie deeply in that **data is a pivotal asset** for the operation of these companies and, at the same time, such data is very sensitive and subject to very stringent legal requirements (e.g., LOPD, GRPD).

To address this problem, major Cloud providers already **comply with certifications** that enable them to run services subject to regulatory compliance (e.g., ISO9001, ISO27001, PCI DSS). The efforts to comply with such certifications are significant, while maintaining them require a constant improvement process from Cloud providers. In fact, this trend is not only followed by public Clouds, but also by private ones that have identified a relevant value added to offer services with high compliance standards. And following this trend, "EU-SEC Financial Pilot" defines different options to deploy Financial Information Sharing services, deploying them using an off-the-shelf SaaS solution (i.e. over Fabasoft cloud) or building it using a commercial IaaS solution with open source components (i.e. Amazon Web Services and Nextcloud[1]).

Unfortunately, the devil is in the details: to obtain or maintain a certification, Cloud providers are subject to "periodic and human-based" auditing processes. Traditionally, this methodology

[1] https://nextcloud.com/

is known as **"point-in-time" auditing**. As one can easily infer, this leads to two major problems: i) the **lapse of time between audits** could be potentially large (e.g., months, 1 year), which means that there is no guarantee that a given provider continuously complies with a set of requirements; and ii) it is **hard for an auditor to inspect all the potential sources of information** to detect legal violations, due to the lack of automation in the auditing process.

The inherent problems of point-in-time auditing are still an important source of uncertainty and lack of trust for many companies that are reluctant to move to the Cloud.

1.1 OBJECTIVES AND SCOPE

The objective of this deliverable is to demonstrate the EU-SEC framework for continuous auditing scenarios. In particular, we aim at demonstrating that the EU-SEC framework can remove many of the compliance-related problems that prevent companies under stringent regulations to move data and computing services to the Cloud.

Finally, to this end, this deliverable present a use-case application related to the daily operation of a financial institution, and in particular, of a bank (CaixaBank): the exchange of sensitive financial documents among banks and regulatory authorities. To achieve this objective, the deliverable reports the outputs of the first phases of the WP5 (Figure 1-1), defining the pilot scope and the technical architecture for the continuous auditing of this application, as well as the interactions among all the EU-SEC framework components.

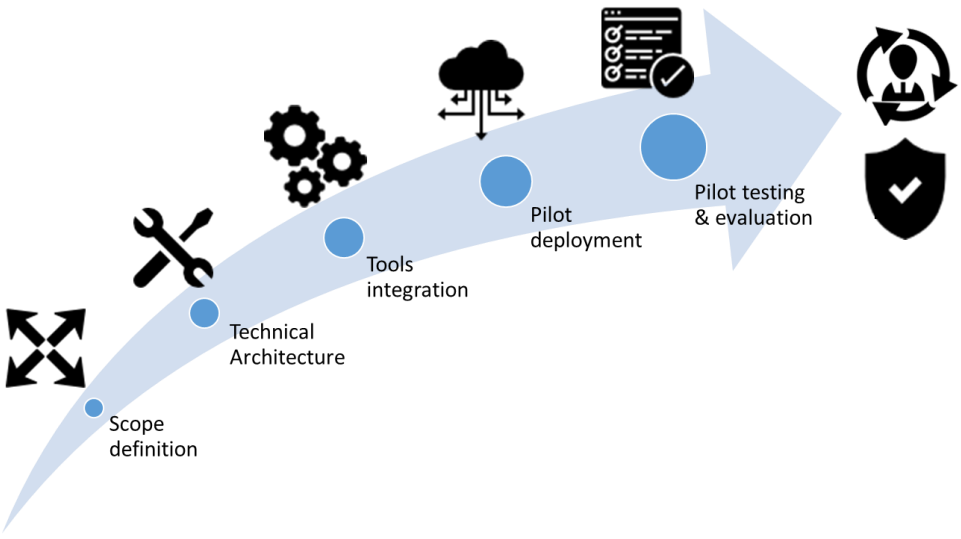

Figure 1-1 WP5 Roadmap: Pilot phases.

1.2 ORGANIZATION OF THIS WORK

This deliverable is organized in three main parts.

The first part is related to the use-case description and pilot architecture. In Section 0, we describe the use-case that motivates the pilot (financial information sharing, FISH). Section 3 defines at high-level the business requirements of CaixaBank in order to run the FISH application in the Cloud. In Section 4, we provide the architecture of the pilot, as well as the deployment decisions and roles of each of the EU-SEC framework components.

In the second part of the deliverable, we describe in depth how each EU-SEC component has been used to meet the use-case requirements. Section 5 translates the high-level requirements of CaixaBank into controls from the Cloud Control Matrix (CCM). This is a necessary first step to understand which controls can be automatically checked by the continuous auditing tool. Section 6 specifies the technical definition of the EU-SEC Continuous Auditing (CA) API, which defines a set of methods to test the controls aforementioned in previous section. In Section 7, we provide a description of the implementation of controls in Clouditor. Next, we describe how the evidences from the continuous auditing are securely stored (Section 8). We describe the role of auditors when inspecting the results of the continuous auditing evidences on our use-case application (Section 9). Finally, we conclude in Section 10.

1.3 WORKPACKAGE DEPENDENCIES

In the following, we describe the dependencies of deliverable with other work packages of this project. As can be observed in Figure 1-2, this deliverable depends on the security and privacy requirements already discussed and defined in WP1, the continuous auditing certification scheme defined in T2.2 and the on-going tasks of WP3, in charge of the associated implementation of tools for continuous auditing.

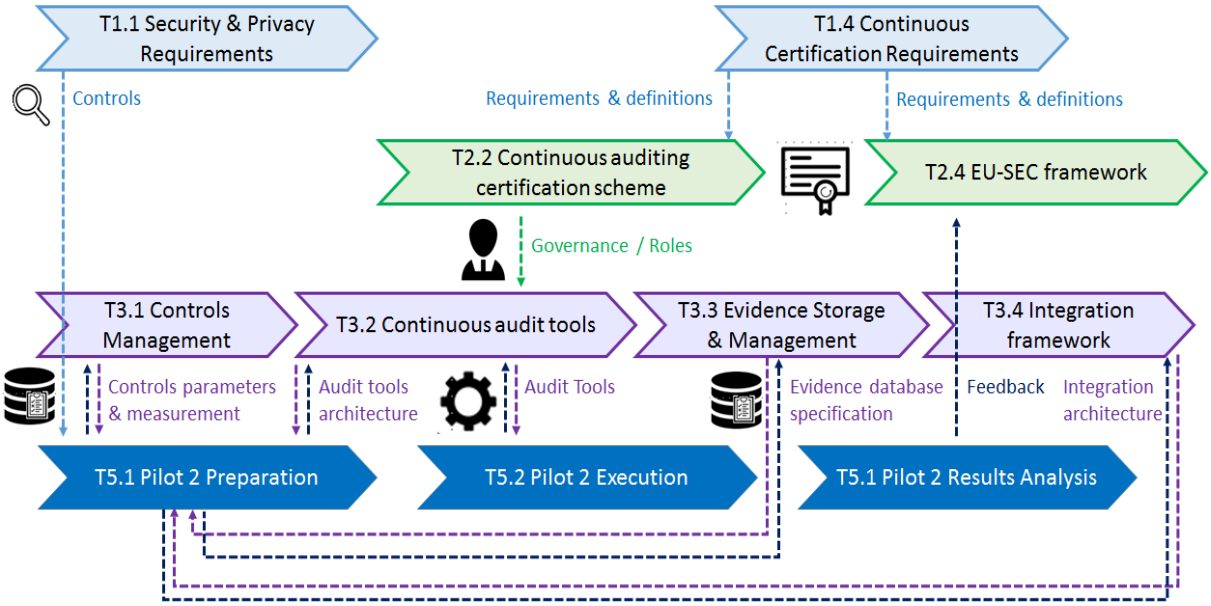

Figure 1-2 WP5 tasks dependencies.

2 PILOT DEFINITION

2.1 FINANCIAL INFORMATION SHARING (FISH)

Regulatory authorities, such as the Bank of Spain (BDE) or the European Banking Authority (EBA), increasingly require financial institutions (e.g., banks, insurance companies) **to report information on their activity or the activity of specific customers**. A motivation for such a strict reporting activity is the interest that many international organizations and countries have in the early detection and prevention of terrorism, money laundering and fraud, among other problems. Note that all European financial institutions are required to comply with such a reporting policy. Therefore, the problem goes beyond the domain of a specific institution or regulatory authority.

Essentially, the reporting activity among financial institutions and regulatory authorities implies **financial information sharing (FISH)**. Based on the experience of CaixaBank, while data sharing is mainly related to documents, there are also other types of information flows that could be shared among financial institutions and regulatory authorities (e.g., pictures, email conversations).

Moreover, the reporting activity could be complex to manage as is **neither unidirectional nor one-to-one**. For example, when a regulator claim a financial entity to report about specific security aspects or incidents occurred, the interaction usually involves several steps and exchanges of sensitive messages and documents in both directions. At some point, the exchange of information could even involve other financial entities when an incident affected more than one entity. This exchange of information is becoming increasingly complex.

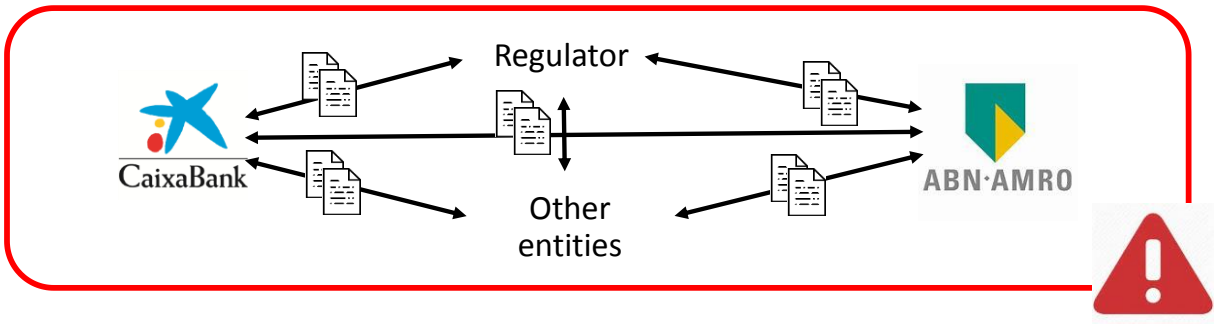

Figure 2-1 Financial Data Sharing (FISH) scenario between Financial Entities and Regulators.

First, a solution to the current situation requires an information sharing service that enables both regulatory authorities and financial institutions to share information. Second, information sharing may involve multiple entities; for instance, a regulatory authority may cooperate with various financial institutions at the same time to investigate the activity of a particular customer, for which is necessary a suitable service that enables data sharing and collaboration. While one may think that a simple in-house document repository per financial institution could satisfy this need, it is clear that it is not practical for regulators when considering a large number of financial institutions and/or multi-party collaborations. Even worse, financial institutions will probably be reluctant to store their sensitive information in the repositories of each other. **This calls to centralize or "cloudify" the FISH service for efficiency and practicality.**

2.2 LACK OF A CONTINUOUSLY AUDITED FISH SERVICE

A **data sharing service in the Cloud** may be a solution to the financial information sharing problem. This would help to i) centralize information sharing among regulatory authorities and financial institutions, ii) enforce security policies and fine-grained access control to reporting information, iii) enable advanced collaboration and sharing functionalities on information.

However, the information shared among regulatory authorities and financial institutions is very sensitive and it is subject to **very strict regulations** (e.g., ISO/IEC 27001, LOPD, GRPD). Unfortunately, the main roadblock today is that there is **no public Cloud provider (e.g., AWS, Microsoft Azure) that offers guarantees on the "continuous" compliance** of such regulations on deployed applications, even though some of them claim to be compliant via "point-in-time" certifications. For instance, let us image a requirement defining that information shared among regulatory authorities and financial institutions should be stored always inside the physical borders of the EU. However, there is no mechanism to ensure that this condition is always enforced by the Cloud provider.

In essence, the core of the problem is the **lack of a continuous auditing service** that verifies that the Cloud provider running the information sharing service actually complies with the required regulation. Solving this problem would become an important goal not only for the CaixaBank use-case, but also for the European regulatory ecosystem.

3 CAIXABANK'S RISK ASSESSMENT

In order to stablish a list of minimum security requirements that a Financial service should comply with when deploying this service in a CSP, CaixaBank has carried out an internal evaluation of the risks associated to the Cloud, and from them and the level of assurance needed to reach by the organization we would get a set of requirements needed to demonstrated that the pilot would achieve a satisfactory level of security

The Risks of Cloud Computing have been developed in an internal CaixaBank's document called: "Guia de Seguridad para el Cloud Computing" with CAIXABANK's internal reference: "CBK-GUI-001".

3.1 INTRODUCTION

Financial Organizations in general and Banks in particular are often a target of attacks, these attacks can come from external threats as well as from internal means of fraud. For this reason security is a priority when assuming new challenges and changes of the Technological infrastructure of the company.

Cloud Computing is not necessarily more or less secure within the current environment, in the same way as any other technology, Cloud is creating new risks as well as new opportunities. In some cases, to adopt Cloud is providing an opportunity to redesign old applications and infraestructures and in this way to comply with new security requirements, in other cases, the risk in moving sensitive data or applications to an emergent infrastructure could represent a non acceptable risk for the company.

3.2 TYPES OF RISKS

- **Loss of control:** In the case of Cloud infrastructure, the client gives necesarilly control to the CSP in some aspects which can affect security. If SLA's (Service Level Agreements) are not correctly including the client's requirements, it could represent a security risk.

  Moreover, the CSP can externalize some parts of the service to third parties (unkown provideers), who are not offering the same level of assurance as the CSP, in consequence the original terms and conditions can change.

This loss of control can have a potentially serious impact in the company strategy and its capacity to comply with its mission and objectives. Could represent a risk of compliance with the basic security requirements such as: Confidentiality, integrity and availability of data, a detriment of quality of service, apart from a lack of regulatory compliance.

- **Control of user accounts:** It is necessary to implement mechanisms in order to assure a basic control of user accounts, key questions such as creation and removal of user, profile management, credentials and passwords, user authentication, etc.. are in CSP hands.
- **Regulatory Compliance and Data Privacy:** In terms of regulatory compliance, the investment to be certified against industry standard certifications or regulatory audits could be put in risk if the service is moved to the Cloud if:
  - The CSP can not present evicences to demonstrate that is compliant with all the necessary requirements
  - The CSP doesn-t allow client's audits or have not logs or evidences of compliance.

  In the particular case of the data Privacy protection and to be able to follow with the European Reglament (GDPR) it is necessary to demonstrate to the final client (who is the owner of the data) that the data is managed in a proper way by the CSP following all the privacy requirements included in the Reglament.

  One aditional problem is appearing in the case of data transfer, for example, between Federated Clouds, because GDPR is restricting international movement of personal data.

3.3 CAIXABANK PILOT REQUIREMENTS

At this point, we have defined the application that will constitute the financial pilot for continuous auditing in EU-SEC. In this section, we describe the specific auditing requirements that CaixaBank has in order to use a financial information sharing service in the Cloud.

Prior describing the requirements of CaixaBank, let us elaborate on the different types of requirements that we may encounter in the proposed pilot. Concretely, we identify the following types of auditing requirements:

- **Platform auditing requirement**: Auditing requirements related to the platform are those for which the auditing service should monitor and analyze metrics related to the instances running applications (VMs, containers). For instance, if a requirement for a service is to store data within EU borders, the auditing service should monitor where data is stored to confirm that this requirement is being fulfilled.

- **Application auditing requirement**: There requirements for which the application itself is responsible to comply. For instance, an application may be responsible to encrypt files prior storing them. Thus, the auditing service should analyze if data has been encrypted (e.g., detect headers, check entropy of information) and alert otherwise.
- **Client auditing requirement**: In our pilot, there are requirements that are related to the client, i.e., CaixaBank. One of them is that evidences should be stored not only in a European trusted Cloud, but also locally in-house to have a local copy for analysis and security purposes.

We believe that the EU-SEC framework could be able of accommodating the implementation of controls for many requirements falling into any of these categories.

Once understood the types of requirements in our pilot proposal, we next enumerate the specific requirements that CaixaBank defines for the FISH pilot proposal:

- **Data location (Platform)**: The location of all sensitive data and its usage by applications and databases should be known. Moreover, all data should be located within European Economic Space.
- In practice, satisfying this requirement in our pilot proposal means that we should continuously monitor all the FISH instances running in a Cloud provider. On the one hand, we may need to collect the location information from each instance that the Cloud itself provides via its API, if it is available. On the other hand, we should collect an alternative location metric (e.g., latency based measurements) of all FISH instances. Such information would help us to infer whether the FISH instances and their data are retained within EU domains, and well as if the Cloud provider tells the truth.
- **Encryption (Application)**: All data should be encrypted both at rest and in transit (AES-256). Cryptographic key management policies and procedures should be defined. In our pilot, this means that we may need to inspect both the connections and the data stored by an FISH application to infer whether data has been encrypted or not.
- **Identity federation (Platform/Application)**: Strong authentication of admin users. Access control and admin profiles should be defined.
- The fulfillment of this requirement may need the auditing service to inspect the logs from both the connections to the instances where the application is running, as well as the administration logs of the application itself. With such information, we could gather evidences of whether a non-authorized access has occurred either at the platform level (instances) or within the FISH application domains.

- **Critical logs in SIEM (Client)**: All monitoring and evidences logs should be stored in CaixaBank infrastructure.
- CaixaBank requires all logs from monitoring an evidences to be stored not only at the control side of the continuous auditing pilot, but also in-house. This requirement needs to be fulfilled due to security and analysis needs of the bank.

4 PILOT ARCHITECTURE AND DEPLOYMENT

We propose a solution for the FISH service, as a feasible CaixaBank use-case pilot in EU-SEC. The architecture for this "Financial Pilot" of EU-SEC project is depicted in the figure below. In what follows, we describe the pilot architecture and user interactions, according to the requirements of CaixaBank.

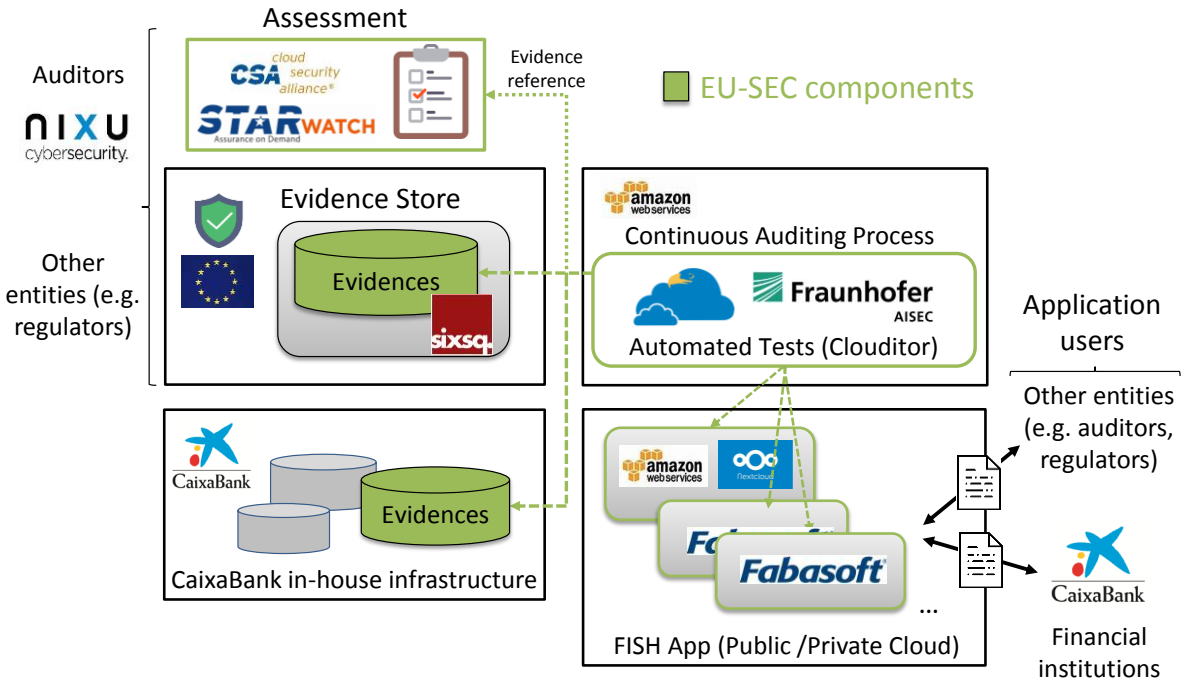

Figure 4-1 EU-SEC Pilot 2 (Continuous Auditing) infrastructure.

In terms of architecture, we observe several entities that form our pilot: the Continuous Auditing Process, the FISH Service, the Evidence Store, the CaixaBank In-house Infrastructure, the Brokerage and Deployment component, and the Assessment component.

FISH Service to Audit (Fabasoft): Public and Private Cloud providers claim to comply with certain service terms and certifications, which are required depending on the application at hand. To certify this, in our pilot we aim at performing continuous audit of the FISH application. In this sense, a candidate application to be audited is Fabasoft, as it would allow financial information sharing among financial entities and regulatory authorities.

In our pilot, we will consider continuous auditing of both instances running the application (VMs, containers) and the application itself. **We will deploy and evaluate two options for providing the FISH Service.** On the one hand, **a specific application is set up over the Fabasoft cloud in a Software as a Service (SaaS) model**. On the other hand, the FISH service can also be built using **a commercial solution with open source components such as Nextcloud**. To this end, on both cases we will benefit from the APIs that some providers offer in their platform that help clients to monitor information about the state of deployed applications. Such APIs may be used to verify that the terms of service or service agreements are actually respected. Therefore, testing these two different approaches the pilot aims at validating the potential adoption of the solution (overall continuous audit infrastructure and API) allowing entities to use it by means of their own or third/party applications in a SaaS model or building up a service over their facilities or in a IaaS/PaaS cloud model.

The following subsections explain how the FISH service is deployed through the different options.

4.1 DEPLOYMENT OF FISH SERVICE WITHIN THE FABASOFT CLOUD (OPTION 1)

The FISH service is a collaborative space for sharing of and working with documents and critical information. It has several critical functional and non-functional requirements:

- Restrictive access right management
- Seamless availability of documents and logging of activities
- Tracking of changes and robust verification of document integrity

These requirements can be met by using the Fabasoft Cloud and for the pilot in WP5, a Virtual Development Environment (VDE) is set up, which brings all the benefits of the Fabasoft Software as a Service (SaaS)[2].

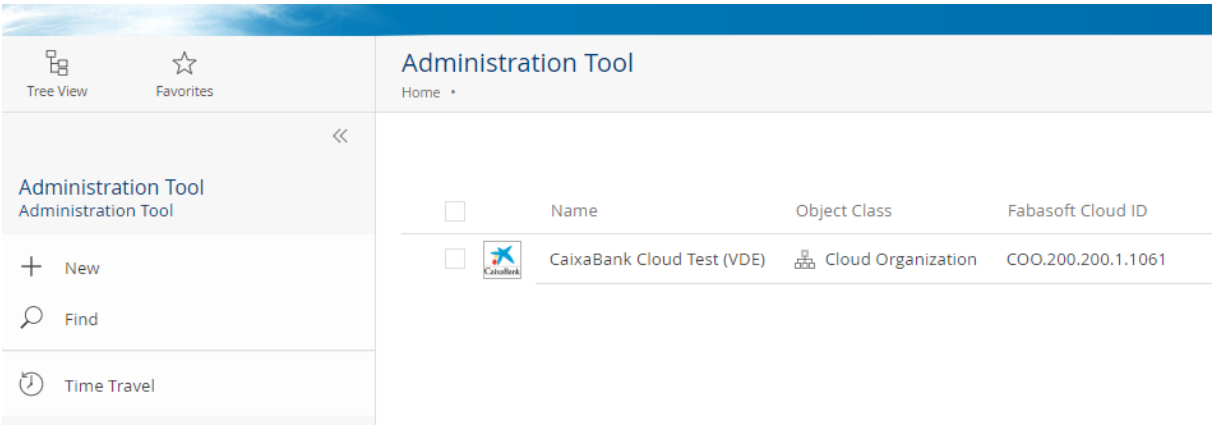

Figure 4-2 VDE set up in Fabasoft Cloud.

For testing purposes, we can administrate the cloud organisation *CaixaBank Cloud Test (VDE)* in the way we need it for the pilot:

- Add Users
- Add / Change Teams
- Manage access rights
- Create Teamrooms with different access rights
- Upload data / information
- While using all certified Fabasoft Cloud capabilities
    - Mobile and location-independent access on company data via browser or apps
    - Control documents across corporate and national borders
    - Workflow management for intern and external business processes
    - Easy and flexible customising by the customer (e.g. modeling and managing approval processes)
    - Seamless versioning and full traceability through the time travel function

In short, we can work with a demo version of an organisation using the application under test. As a basic configuration, the demo version has

- Fifteen Users (one being the owner of the Organisation)
- Four Teams (Managing Board, Consultants, Marketing & PR, Regulation Affairs)
- Four Teamrooms (Internal Communication, External Communication, Customer Communication, Regulator Communication)

[2] https://www.fabasoft.com/en/products/fabasoft-cloud

- Teams have different access rights to those Teamrooms (Marketing & PR does not see Regulator Communication for instance)

The Cloud VDE is available to the consortium via s specific teamroom (Figure 4-3).

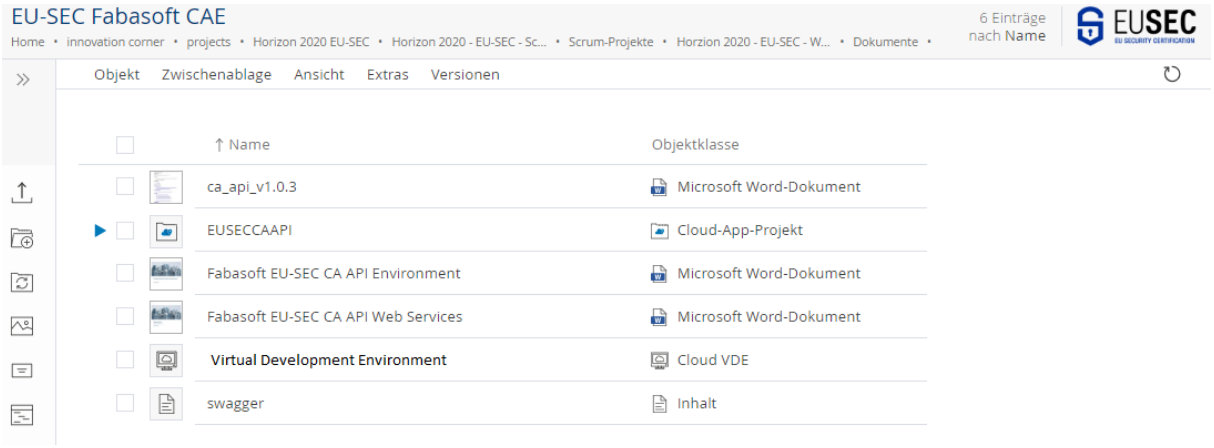

Figure 4-3 WP5 Teamroom for managing VDE

4.2 DEPLOYMENT PROCESS OF FISH UNDER AWS UTILIZING NEXTCLOUD (OPTION 2)

From a technical perspective the FISH service is a file and data sharing application and thus it also possible to deploy an Open-Source based application such as Nextcloud on a public IaaS provider, to facilitate the FISH use cases. This choice introduces set of system requirements from Nextcloud to run:

- Red Hat Enterprise Linux 7 / Ubuntu 16.04 LTS
- MySQL/MariaDB
- PHP 7.0, 7.1 or 7.2.
- Apache 2.4 with mod_php
- At least 512 MB RAM

Those requirements are meet by a variety of bare-metal setups as well as IaaS cloud offerings. For this pilot we've decided for Amazon Web Services to deploy the application due to the cloud nature of this project as well as compatibility reasons with clouditor. Since Nextcloud demands a execution environment, a database as well as storage this pilot runs on:

- a EC2 VM instance

- a RDS Database service providing a MySql instance

additional EB storage with encryption Nextcloud was installed the following way:

1. Creating an EC2 instance with CentOS 7
    a. Assigning of an elastic IP to the EC2 instance
    b. Installation of Apache 2
    c. Installation of PHP and required Apache modules
    d. Adjustment of SELinux with necessary rules
    e. Creation of self-issued SSL certificate
    f. Adding redirection rules to https
2. Creating a RDS instance with MySql
3. Creating a new EBS volume with encryption.
    a. Adding the volume to the mount table of the EC2 instance
4. Installing Nextcloud

To meet the extended information needs for continuous auditing it's necessary to extend Nextcloud logging capabilities. In order not to interfere with Nextclouds own code integrity checks changes to the core logging system haven't been made. The pilot alternatively provides the FISH functionality via a Plugin with its own logging capabilities necessary for continuous auditing.

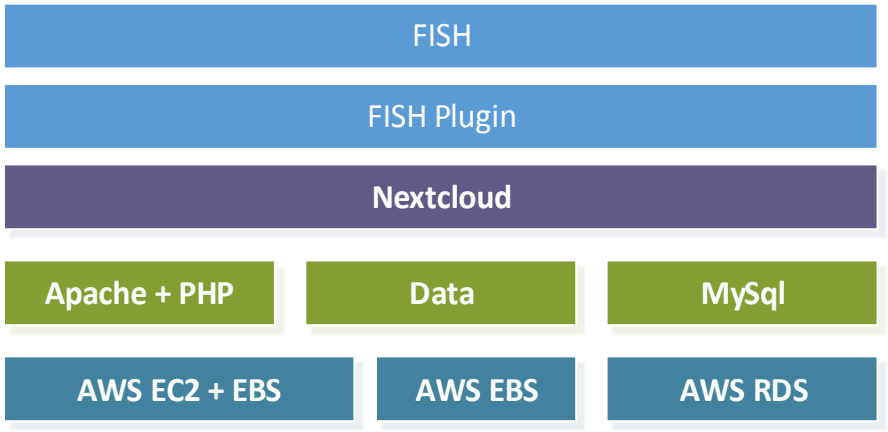

Figure 4-4: Fish deployment under AWS.

4.3 DEPLOYMENT OF CLOUDITOR

This financial pilot aims at executing a continuous auditing process of an application deployed in a Cloud. To perform the continuous auditing process, **we propose to use Clouditor** in order to

- monitor a set of metrics for instances of a given application using for example the provider's API or, if implemented, the EU-SEC Audit API.
- export the evidences to an external repository for further analysis and auditing.

Therefore, it is important to note that the pilot decouples the collection of evidences from its analysis, which may apply to other applications apart from a financial information sharing application. Within the pilot we chose to deploy the Clouditor into the same Cloud environment (AWS) as the Nextcloud-based FISH-variant. However, this deployment will monitor both, the Nextcloud-based and Fabasoft-based variation of FISH.

Deploying it directly into the same Cloud environment serves two purposes. For the Nextcloud-based FISH, it simulates that a provider (i.e. the provider of the Nextcloud service) is installing the audit tool directly into its own environment, i.e. to satisfy certain security requirements. For example, using AWS IAM roles, certain API privileges can be assigned to the virtual machine running Clouditor without having to give out credentials to the Cloud APIs to third parties or to remote locations. The provider can then chose to share the gathered information with third parties, such as auditors either directly using the Clouditor dashboard or via the Starwatch registry (see chapter 4.5).

For the Fabasoft-based FISH variant, this deployment represents the case of a hosted, managed installation of a Clouditor. In this case, the provider, i.e., the provider of FISH-Fabasoft, does not own the Clouditor instance, but has contractually assigned this to a third-party. In this case, API credentials need to be transferred to a third-party and remote locations and thus, proper encryption of those credentials in transit and rest need to be established. In the pilot, this is realized using standard AWS features such as encryption of volume storage and industry standards such as TLS.

The Clouditor itself can be installed on any Linux-based machine, ideally using Docker. Additionally, Kubernetes templates are also available, if a deployment into a Kubernetes cluster is preferred. It three different components:

- The Clouditor Engine, which executes checks and collects evidences

- The Clouditor Explorer, which gathers additional facts about the Cloud environment the engine is running on. Deployment of the Explorer is optional
- The Clouditor Dashboard, which displays results of checks and claims in a web-interface. Deployment of the Dashboard is also optional

A database is needed to store the state of the Clouditor, such as the configured checks, the certification objects, etc. Currently, the Clouditor only supports MongoDB as a database back-end, while the gathered evidences are forwarded to the Evidence Store.

4.4 DEPLOYMENT OF EVIDENCE STORE

We assume the existence of a European Certified Cloud which can be trusted by both financial institutions and regulatory entities (e.g., its figure may resemble a certification authority). This infrastructure is intended to support the **storage of evidences from applications running and being continuously audited**. The need for this infrastructure to be trusted by alternative means (e.g., physical auditing, certifications) is to break the recursive problem that automated auditing poses: i.e., *who audits the infrastructure storing evidences?* This kind of chicken-and-egg problem is hard to solve, so a way of breaking it is to rely on a trusted Cloud to run only the control of applications running in other non-trusted Clouds.

Note that the evidence store is subject to strong security requirements. For instance, it must ensure that evidences can only be written once and not modified afterwards, in order to avoid fraud or counterfeit of auditing evidences. Moreover, we propose to run in the European Trusted Cloud an evidence store that enables auditors to autonomously access the evidences in order to detect violations in the compliance of regulations in audited applications. If the Cloud provider claims to respect a regulation that specifies to keep an application's data within EU borders, then the auditor can afterwards act accordingly. Note that the evidence store could be also used by other parties related to a specific applications that are concerned by the auditing process. Besides, having an evidence store in the Cloud makes life easier for auditors to access and inspect the compliance of regulations of multiple applications.

For this pilot the evidence store will re-use an existing infrastructure from one of our trusted partners within the EU-SEC project, SixSq. One of SixSq's functions is to act as a Cloud Brokerage Service, thus having the perfect foundation for an audit portal where not only the auditor can deploy their continuous auditing tools, but also later on consult and manage the resulting evidence. One can also envision a future scenario where some of the evidence (only

the public one) can be made available to regular cloud customers, in a multi-cloud environment, so that they can optimize the selection of cloud providers.

SixSq's existing infrastructure is deployed in a Swiss Trusted Cloud provider, Exoscale, and all the pilot relevant services (database, interface and web portal) are located in their Geneva's data centre.

These three services are open-source, so in fact, they can be deployed anywhere in case a more private setup is desired/required.

Due to the requirements of Caixabank company, **a copy of the monitoring information and evidences must be kept** in CaixaBank facilities as well. Therefore, Caixabank will collect a copy of the evidences and store in its own in-house infrastructure. The CaixaBank copy of the continuous auditing evidences will be used for security purposes jointly with other sources of information collected within the company.

4.5 DEPLOYMENT OF ASSESSMENT COMPONENT

The last item of our pilot is related to the assessment of the controls to be continuously audited. For this reason, we resort to Starwatch to enable auditors to keep track of which controls are being enforced in a simplified manner. More details specified how it is deployed and how auditors can interact with this component are described in section 9.

5 DEFINITION OF CONTROLS

In this section, we identify the set of controls derived from the business requirements from CaixaBank. Moreover, we relate the controls with the ones defined in the Cloud Control Matrix (CCM), which is part of our work in WP1.

Table 5-1 Required controls.

Data Location	Type	Control	CCM Code
Local VM data	Platform	Location of all sensible data and its usage by applications and databases should be known	CCM-GRM-02
Persistent Data Storage	Platform	All data should be located within European Economic Space	CCM-STA-05

Encryption	Type	Control	CCM Code
Encryption on data transfers and data at rest	Application	All data stored on Cloud should be encrypted in rest and in transit	CCM-EKM-04
Key management	Application	Cryptographic key management policy and procedures should be defined.	CCM-EKM-02
Secure cyphers	Application	AES-256 should be used and only CaixaBank should be the owner of cryptographic keys	CCM-EKM-04
Identity Federation	**Type**	**Control**	**CCM Code**
VM access control	Platform	Identity administration federated to the administrator of CaixaBank	CCM-IAM-12
Application authentication	Application	Strong authentication of admin users	CCM-IAM-12
Application access control	Application	Access control and admin profiles should be defined	CCM-IAM-12
Evidence Security	**Type**	**Control**	**CCM Code**
Store evidences in CaixaBank	Client	All critical logs should be send to the SIEM of CaixaBank	CCM-IVS-01

Please, note that these requirements have been agreed with the cloud security division here in CaixaBank; these are minimal requirements for other applications to run in the cloud. If EU-SEC can meet these requirements in this pilot, we can then further extend the controls to comply with GDPR as a second step.

6 CONTINUOUS AUDITING API

This section specifies the technical definition of the EU-SEC Continuous Auditing (CA) API, which defines a set of methods to test the controls aforementioned in previous section. The list of methods provided by EU-SEC CA API are:

CaApiDataLocation

- GET /{scope}/datalocation/{objectId}/storage/

CaApiEncryption

- GET /{scope}/encryption/{objectId}/

CaApiIam

- GET /{scope}/identityfederation/admins/
- POST /{scope}/identityfederation/data/access
- GET /{scope}/identityfederation/{userId}/logins
- GET /{scope}/identityfederation/{userId}/auth
- GET /{scope}/identityfederation/{userId}/groups

CaApiScope

- GET /scope/

6.1 CAAPIDATALOCATION

6.1.1 DATA LOCATION STORAGE

```
GET /{scope}/datalocation/{objectId}/storage/
```

Returns persistence information for a particular data object by its Id (*getDataLocationStorage*). Depending on the kind of storage this call returns an identifier of the particular storage entity. E.g. database name, RDS id, Harddrive, SMB location etc. If stored on multiple services all are returned. This requires each logical object to be traceable to its physical persistence capabilities. It is based on CCM-GRM-02. It has to be remarked that the location obtained is always regarding to the data storage geographical location when stored in the cloud and potential people privacy issues are not affected in any occasion.

PATH PARAMETERS

- *objectId (required):* ID of data object to return.
- *scope (required):* Scope of service.

RETURN TYPE

- LocationStorageResponse.

RESPONSES

- Successful operation LocationStorageResponse (200).
- Invalid input (405).

PRODUCES

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json.

EXAMPLE DATA

```
{
  "storages" : [ {
    "uri" : "i-0434c5582f2853d0c",
    "type" : "service",
    "description" : "AWS EC2 insctance"
  }, {
    "uri" : "vol-04b6088c76eb68a73",
    "type" : "service",
    "description" : "AWS EBS instance"
  }, {
    "uri" : "jdbc:mysql://192.168.0.10/SuperDB",
    "type" : "database"
  } ]
}
```

6.2 CAAPIENCRYPTION

6.2.1 ENCRYPTION INFO

```
GET /{scope}/encryption/{objectId}/
```

Retrieves the encryption info of an object (*getEncryptionInfo*). Propper interpretation has to be performed by the audit tool. Based on CCM-EKM-04.

PATH PARAMETERS

- *objectId (required):* ID of either objectId on SaaS level or storeageId on lower level
- *scope (required):* Scope of the service.

RETURN TYPE

- EncryptionStorageResponse.

RESPONSES

- successful operation EncryptionStorageResponse (200).
- Invalid input (405).

PRODUCES

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json.

EXAMPLE DATA

```
{
  "keyOrigin" : [ {
    "keyOriginUri" : "hsm://secret.datacenterX",
    "type" : "hsm",
    "description" : "Supersecure HSM"
  }, {
    "keyOriginUri" : "smb://key.pem",
    "type" : "localKeyFile",
    "description" : "Used for AES-256 enc."
  } ]
}
```

6.3 CAAPIIAM

6.3.1 ADMINISTRATORS

```
GET /{scope}/identityfederation/admins/
```

Returns a list of administrators *(getAdmins*). Reads out all administrators of the application and returns them. Based on CCM-IAM-12.

PATH PARAMETERS

- *scope (required):* Scope of the service.

RETURN TYPE

- inline_response_200.

RESPONSES

- Successful operation *inline_response_200* (200).
- Invalid input (405).

PRODUCES

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json.

EXAMPLE DATA

```
{
  "admins" : [ "adminXYZ", "root", "caixaAuth" ]
}
```

6.3.2 ACCESS

```
POST /{scope}/identityfederation/data/access
```

Checks whether a user has a certain access to an object (*getObjectAccess*). Based on CCM-IAM-12.

PATH PARAMETERS

- *scope (required):* Scope of the service.

BODY PARAMETER

- request object (required).

RETURN TYPE

- inline_response_200_3.

RESPONSES

- Successful operation *inline_response_200_3* (200).
- Invalid input (405).

PRODUCES

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json.

EXAMPLE DATA

```
{
  "allowed" : true
}
```

6.3.3 USER ACCESSES

```
GET /{scope}/identityfederation/{userId}/logins
```

Returns a list of the last logins of a user (*getUserAccesses*). Whenever a user logs in into the application this kind access gets logged. This call returns the last accesses of a particular user based on existing logs. Based on CCM-IAM-12.

PATH PARAMETERS

- *userId (required):* ID of user.
- *scope (required):* Scope of the service.

QUERY PARAMETERS

- *from (optional):* from date.
- *limit (optional):* Limits the number of retuned values.

RETURN TYPE

- *inline_response_200_2*

RESPONSES

- successful operation *inline_response_200_2 (200)*.
- Invalid input (405).

PRODUCES

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json.

EXAMPLE DATA

```
{
  "loginTimes" : [ "2005-08-15T15:52:01+0000" ]
}
```

6.3.4 AUTHENTICATION TYPE

```
GET /{scope}/identityfederation/{userId}/auth
```

Returns the authentication type of a user (*getUserAuthType*). Reads out a particular users authentication settings and returns them (e.g. password, two-factor). Propper interpretation has to be performed by the audit tool. Based on CCM-IAM-12.

PATH PARAMETERS

- *userId (required):* ID of user.
- *scope (required):* Scope of the service.

RETURN TYPE

- *AdminAuth.*

RESPONSES

- successful operation *AdminAuth* (200).
- Invalid input (405).

PRODUCES

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json.

EXAMPLE DATA

```
{
  "description" : "description",
  "type" : { }
}
```

6.3.5 GET USER ORGANISATION

```
GET /{scope}/identityfederation/{userId}/groups
```

Returns the groups of a user depending on the implementation a group can be e.g a unix group, organisation, role etc. (getUserOrganisation). Based on CCM-IAM-12.

PATH PARAMETERS

- *userId (required):* ID of user
- *scope (required):* Scope of the service

RETURN TYPE

- *inline_response_200_1*

RESPONSES

- successful operation *inline_response_200_1* (200)
- Invalid input (405)

PRODUCES

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json.

EXAMPLE DATA

```
{
  "groups" : [ "root", "awsEc2Full", "users" ]
}
```

6.4 CAAPISCOPE

6.4.1 SCOPE

```
GET /scope/
```

Returns all scopes of the cloud service (getScope). The scope corresponds often with the layers of the cloud service architecture like IaaS, PaaS, SaaS.

RETURN TYPE

- *ScopeResponse*

RESPONSES

- successful operation *ScopeResponse* (200)
- Invalid input (405)

PRODUCES

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

EXAMPLE DATA

```
{
  "scopes" : [ "", "" ]
}
```

7 IMPLEMENTATION OF CONTROLS

In this section, we identify the set of controls derived from the business requirements from CaixaBank. Moreover, we relate the controls with the ones defined in the Cloud Control Matrix (CCM), which is part of our work in WP1.

Table 7-1 Required controls implementation

Data Location	CCM Code	Implementation in Clouditor	Automated
Local VM data	CCM-GRM-02	• Checking the location of the VM, databases and storage via GeoIP tools. • Checking the location of the VM, databases and storage via provider-specific APIs and using the EU-SEC Audit API.	Y
Persistent Data Storage	CCM-STA-05	See CCM-GRM-02	Y
Encryption	**CCM Code**	**Implementation in Clouditor**	**Automated**

Encryption on data transfers and data at rest	CCM-EKM-04	• Checking TLS quality on transport layer. • Checking used encryption algorithms for storage at rest, i.e. using the Audit API as well as platform-specific APIs (S3, EBS).	Y
Secure cyphers	CCM-EKM-04		
Key management	CCM-EKM-02	• Checking the location and quality of keys stored in the Cloud using APIs, such as Amazon KMS.	Y
Identity Federation	**CCM Code**	**Implementation in Clouditor**	**Automated**

VM access control	CCM-IAM-12	• Checking Access Control lists and enforcement on multiple levels, i.e.
		  ○ Network ACLs to access the VM.
		  ○ ACLs defined in the application to access data objects.
		• Checking for inactive users, expired passwords and good password policies.
			Y

Application authentication	CCM-IAM-12		
Application access control	CCM-IAM-12		
Evidences in SIEM	**CCM Code**	**Implementation in Clouditor**	**Automated**
Store evidences in CaixaBank	CCM-IVS-01	• Not implemented at the moment. Clouditor does not ingest evidences directly into CaixaBank's SIEM. • Evidences will be collected and stored in CaixaBank's in-house storage service for later further analysis when necessary.	N

8 STORAGE AND MANAGEMENT OF EVIDENCES

As described above, the evidence store will re-use SixSq's existing infrastructure. More specifically, the relevant software stack for this pilot comprises an ElasticSeach[3] database, an infrastructure management interface based on the CIMI specification from DMTF[4], and Nuvla[5], SixSq's online service broker based on SlipStream[6].

Since all three components are already deployed and in production, all that is required is that the auditor signs up with Nuvla and uses that account to programmatically store evidence in the database, through CIMI.

For the evidence store, a new CIMI resource has been created – *evidenceRecord*. This resource's schema is aligned with the test results coming from the continuous auditing tool – Clouditor:

```
{
   "endTime": datetime,
   "class": string,
   "startTime": datetime,
   "planID": string,
   "updated": datetime,
   "passed": boolean,
   "created": datetime,
   "id": string,
   "acl": map,
   "operations": list,
   "log": list,
   "resourceURI": string
}
```

Apart from these attributes, the *evidenceReport* resource presents an open schema, where additional attributes can be added, as long as they are prefixed with a namespace which has already been registered in the database, i.e. *<namespace>:<attributeName>*. As an example, for a specific type of Clouditor tests called *RandomTest*, Clouditor can optionally add any attributes to the *evidenceRecord* resource as long as the namespace *RandomTest* has been previously created.

[3] https://www.elastic.co/
[4] http://www.dmtf.org/sites/default/files/standards/documents/DSP0263_2.0.0.pdf
[5] https://nuv.la/
[6] http://ssdocs.sixsq.com/en/latest/index.html

Through CIMI, Clouditor can use its Nuvla account to perform Create, Read and Delete operations on the *evidenceRecord* resource. This means that given the proper credentials, Clouditor will be able to store new evidence, read existing evidence and delete specific evidence records. Editing existing evidence will not be allowed.

The existing CIMI implementation exposes a RESTful API which is compliant with any HTTP client Clouditor might use to manage evidence.

To ensure the confidentiality, integrity and availability of the evidence records: all management operations are secured over TLS; only the infrastructure administrators and Clouditor are authorized to access and manage the evidence coming from Clouditor (unless Clouditor specifically sets the *evidenceRecord* ACLs differently); all production software procedures will apply to the evidence store, including the regular data backups, active customer support and periodic system updates.

9 CONTINUOUS CERTIFICATION

In a continuous auditing process, the cloud service provider (CSP) must report on a timely manner wether it is compliant with a set of objectives defined in a certification target, as detailed in D1.4. This reporting is addressed to an "authority" which maintains a public registry of ongoing continuous certificates. As long as the CSP confirms objectives according to the certification target, the certificate is listed by the authority as "valid". If the CSP fails to confirm an objective on a timely manner the certificate status is changed from "valid" to "suspended" by the authority. After a "grace period", if the CSP has still failed to confirm one or more objectives on a timely manner, the authority will change the status of the certificate to "revoked" and it will be removed from the public registry. This whole process is initiated by the CSP, which will need to submit to the authority a set of objectives it is commited to achieve. Note that depending on the certification model, these objectives might also need to be validated by an independent auditor, as detailed in D1.4.

We can summarize the tasks conducted by the authority as follows:

1) Collect certification targets submitted by CSPs.
2) Receive updates form CSPs regarding objectives and verify if they match the initial submitted certification target.
3) Maintain a public registry of ongoing continuous certificates.

The Cloud Secuirty Alliance, a partner in this project, intends to play the role of such an authority in the pilot as well as to establish an operational continuous certification platform in the future. To this end, it will extend STARwatch, its compliance SaaS application, to implement the 3 tasks described above.

More concretely, the STARwatch application will be extended to include:

1) An API endpoint for the submission of certification targets.
2) An API endpoint for the submission of objective status updates.
3) A public registry of ongoing continuous certifications, reflecting the compliance status of participating CSPs (i.e. "valid", "suspended", "revoked").

This section describes these elements in detail.

9.1 PROCESS OVERVIEW

Currently STARwatch is designed to process CAIQ questionnaires, which have the following three-level structure:

- **Level 1**: CCM domain
    - **Level 2**: CCM control
        - **Level 3**: CAIQ question

In the context of the EU-SEC project, STARwatch will be extended to offer a new type of assessment: a continuous assessment. These new type of assessments will also be structured with a three-level structure:

- **Level 1**: CCM domain
    - **Level 2**: CCM control
        - **Level 3**: Objective (i.e. SLO or SQO).

While the mapping between CCM controls (Level 2) and CAIQ questions (Level 3) is standardized, this is not the case for the mapping between CCM controls (Level 2) and Objectives (Level 3) in continuous assessments. This mapping is defined by the user, following industry best practices and the specifics of the system being assessed, in what is called "a certification target". The user must therefore first define "a certification target" before STARwatch can start a continuous assessment.

The STARwatch continuous assessment process can be therefore divided in two phases.

1) Initialization: the Service Provider prepares the continuous assessment privately, defining the certification target and a starting date.
2) Assessment: After the starting data, the Service Provider sends regular updates updating the security status of the Service, according to the certification. STARwatch updates the status of the Service in a public registry accordingly.

In the following, each phase is further detailed:

9.1.1 INITIALIZATION PHASE

The initialization phase describes the private part of the process where the Cloud Service Provider will define the certification target, by specifying:

- The scope of the service covered by the continuous assessment.
- The requirements, SLOs or SQOs that will be continuously assessed.
- The frequency of assessment for each of the selected requirements, SLOs or SQOs.

These elements can be fully specified through the JSON data format defined in deliverable D3.1, which was called "certification objective". For consistency with other deliverables, we will rename that structure as a "certification target".

In the pilot, the Cloud Service Provider will therefore prepare a JSON certification target and upload it to STARwatch. This upload will create a new "continuous" CCM assessment in STARwatch, and will trigger the second part of the process: the assessment phase.

9.1.2 ASSESSMENT PHASE

During the assessment phase, STARwatch will collect assessment data from auditing tools (and human auditors) and reflect this information in a public registry describing the certification status of each assessed Service Provider. A certification status can have three values:

- "valid": All controls, SLOs or SQOs in the "certification target" have been verified at their expected frequency of assessment. The certificate is listed in the public registry.
- "suspended": The assessment contains at least one control that has not been verified within the time interval specified in the "certification target". The certificate is listed in the public registry.

- "revoked": The assessment has been in "suspended" state for too long ("Grace Period"), without satisfactory response from CSP. The certificate is no longer listed in the public registry.

Once we reach the start date defined in a certification target for a service provider, StarWatch will create a corresponding entry in a public registry. At this initial point, the public entry will list the certification status of the cloud service as "valid". This status will remain the same or change depending on the timely provision of assessment data to STARwatch. If the certification status of a service provider becomes "revoked" then the corresponding entry will get removed from the public registry and the process will need to be started again from the initialization phase.

An API enables external tools to submit assessment data to STARwatch. Each submission will notably include:

- A reference to the assessment to which the results pertain.
- A reference to the control, SLO or SQO for which an assessment result is provided.
- A measurement date, describing when the external tool computed assessment result.

We will distinguish the **measurement date** from the **submission date**, the latter referring to the point in time where the data was actually received by STARwatch.

Consider the following notations:

- When the service provider creates a "certification target", he provides a start date we will call $\mathbf{T_0}$.
- Each objective C*i* referenced in a certification target will have a submission period P*i* (e.g. P*i* = 10 days).
- For each obejctive C*i*, we define the *n*th submission interval W*i*,*n* as the time interval starting at T_0+Pi*n and ending at T_0+Pi*(n+1).
- There is a globally defined period G, called the "grace period" (e.g. G = 20 days), which applies to all continuous assessments.

STARwatch will apply the following 6 rules to define the certification state of a service provider.

- **Rule 1**: At time T0 the continuous assessment has a status set to "valid".
- **Rule 2:** For each objective Ci, at least one submission must be made in each time interval Wi,n.
- **Rule 3:** For each submission for objective Ci provided according to Rule 2:

- The answer field provided in the submission must match the expected answer defined in the certification target.
- The measurement date that is provided as part of the submission must also fall within the corresponding time interval Wi,n, according to the period defined in the certification target.

- **Rule 4:** Given an assessment with a "valid" status, if there is one objective Ci for which there is not a single submission that verifies both Rule 2 and Rule 3 during the time window Wi,n, then the status of the assessment is set to "suspended" at the end of the corresponding time window Wi,n. This time marks the beginning of the "suspended" status.
- **Rule 5:** Given an assessment with a "suspended" status, if at one point all questions in an assessment verify Rule 2 and Rule 3, then the assessment is set to "valid" status again.
- **Rule 6:** If an assessment remains continuously with a "suspended" status for a period of time greater than G, then the assessment is marked as "revoked", and is not monitored anymore or displayed in the public registry.

9.2 INTEGRATING CERTIFICATION TARGETS

In deliverable D3.1, we defined a JSON data format that is designed to express a certification target, which is essentially:

- A list of high-level requirements (e.g. control objectives) broken down into objectives (SLOs/SQOs or again control objectives).
- An assessment frequency for each objective.

The data format makes a distinction between objectives that are can be assessed automatically (automated_assessment) versus those that require human intervention (assisted_assessment).

We recall below the general structure of this JSON data structure, referring readers to deliverable D3.1 for details. As a change, we renamed the very first field of the data structure from "certification_objective_id" to "cerification_target_id". The JSON notations used here are the same as in D3.1 and are summarized in Annex 11.1.

```
{
"certification_target_id": <string>,
"start_date": <datetime>,
"end_date": <datetime>,
"subject": {
"organisation": <string>,
"service": <string>,
"scope": <string>,
    },
"assessment":
"type": <string>,
"auditor": <string>,
"authority": <string>
    }
"requirements": [
        {
"requirement_id": <string>,
"requirement_framework": <uri>,
"objectives": [
<assisted_assessment> | <automated_assessment>,
                …
            ]
        },
        …
    ]
}
```

When the certification target JSON file is uploaded, STARwatch will use the data as follows:

Table 9-1 STARwatch process of certification information

JSON property	How STARwatch will process the property
certification_target_id	Ignored as input
start_date	Start date of the continuous assessment
end_date	End date of the continuous assessment
subject.organization	An identifier which must refer to a an organization already registered in STARwatch (i.e. Amazon = 17).
subject.service	The name of the service to be published in the public registry
subject.scope	A textual description of the service and the scope of the assessment.
assessment.type	*Ignored as input*
assessment.auditor	*Ignored as input*
assessment.authority	*Ignored as input*
requirements.requirement_id	Will match the corresponding CCM control identifier (e.g. AIS-01).

requirement.framework	Set to "https://cloudsecurityalliance.org/download/cloud-controls-matrix-v3-0-1/"
requirement.objectives	Will only contain **automated assessments**.

The structure of an **assisted_assessment** is as follows:

```
{
"objective_id": <string>,
"frequency": <duration>,
"type": "assisted",
"asset_name": <string>,
"description": <string>
}
```

In the context of the EU-SEC pilot, we will not use assisted assessments.

The structure of an **automated_assessemnt** is as follows:

```
{
"objective_id": <string>,
"frequency": <duration>,
"type": "automated",
"asset_name": <string>,
"metric": <uri>,
"attribute_name": <string>,
"measurement_parameters": [
        {
"name": <string>,
"type": "number" | "long" | "boolean" | "string",
"value": <number> | <long> | <string> | <boolean>,
        },
        ...
    ],
"result_format": [
        {
"name": <string>,
"type": "number" | "long" | "boolean" | "string",
        },
        ...
    ],
"assertion": <string>
}
```

When the certification target JSON file is uploaded, STARwatch will use the data provided in each automated assessment as follows:

Table 9-2 STARwatch process of certification information (automated assessment)

JSON property	How STARwatch will process the property
objective_id	STARwatch will store this value: tools that submit an update to an assessment will reference this id.
frequency	Used by STARwatch to determine the frequency of assessment.
type	Set to "automated" as defined in D3.1.
asset_name	Used by STARwatch when displaying an continuous assessment in the registry.
metric	Used by STARwatch when displaying an continuous assessment in the registry.
measurement_parameters	Ignored
result_format.name	STARwatch will store this value: tools that submit an update to an assessment will reference this name.
result_format.type	Set to "boolean": tools submitting an update are expected to check weather the SLO/SQO is achieved, thus reporting "true" or "false".
assertion	Ignored

The successful uploading of a JSON certification target in STARwatch will result in the creation of a CCM assessment added to the list of assessments that is under the user's STARwatch license. The corresponding assessment identifier will be provided to the user on the screen (assessment_id). This assessment_id will be necessary for automated tools that will provide updates to the CCM assessment.

9.3 UPDATING CONTINUOUS ASSESSMENTS

The STARwatch application will expose a method enabling tools to submit updates to STARwatch assessments with the following signature:

```
PUT /api/v1/continuous/assessment_update
```

This request must be accompanied with an API key provided in the in the "Authorization" request header, e.g.:

```
Authorization: Token token="ukTvhgtC3xYBt72PhlCRvI5qsQvp"
```

Further details about the API key are provided in Annex 11.2.

The body of the PUT request will have the following structure

```
{
assessment_id: <string>,
objective_id: <string>,
result: <boolean>,
assessed_at: <UTC_time>,
evidence: [              // optional
<string>,
        ...
    ]
}
```

Table 9-3 STARwatch process of certification information (continuous assessment)

JSON property	Description
assessment_id	Refers to the continuous assessment that is being updated. This identifier is obtained after uploading the JSON certification target as described in section 9.2.
objective_id	Refers to the objective that is being updated.
result	A Boolean describing whether or not the objective is fulfilled.
assessed_at	A timestamp describing when the objective was assessed.
evidence[]	An array of pointers to supporting evidence. These strings can be URLs or simply identifiers, if the context is sufficiently clear. STARwatch will not perform any checks on these values or publish them in the public registry, but will display them to the service owner.

10 DISCUSSION AND CONCLUSIONS

In this document, we provided a proposal for a continuous auditing use-case in the financial sector. In particular, we proposed to leverage a financial information sharing service in the Cloud thanks to the exploitation of the EU-SEC continuous auditing framework.

The proposed pilot architecture fulfills a set of requirements to deploy this kind of service in the Cloud while complying with a set of current regulation requirements. Furthermore, as one can infer, such architecture may also accommodate the auditing requirements for many other types of applications, which makes our contributions valuable.

11 ANNEXES

11.1 CONVENTIONS

In the following sections, JSON objects will be described directly in a pseudo language that uses and extends JSON itself, and is mostly self-explanatory. This approach frequently used in the industry (e.g. Google Cloud API[7]) was preferred over more formal approaches such as JSON schema[8] for readability and ease of use. It was used in Deliverable 3.1 as well.

Types

A type is described by a name enclosed between "<" and ">". In addition to the standard JSON types <string>, <number> and <boolean>, we will also use the following base types:

- <long>: A integer number.
- <datetime>: A string representing a UTC timestamp as defined in ISO 8601, including the year, month, day, hour, minute and second, and ending with the 'Z' marker representing UTC time (e.g. 2016-09-29T13:11:43Z).
- <duration>: A duration as defined in ISO 8601, using the extended format P[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss].
- <uri>: A string representing a URI (Uniform Resource Identifier), typically a URL.

Example:

```
{
"name": <string>,
"age": <long>
}
```

In addition to the primitive types above, this specification defines additional objects that are detailed each in their own sub-section. These objects are named with the same convention as above, with a type name enclosed between "<" and ">" (e.g. <assisted_assessment>).

Multiple choices

When an object can take several values or types, this choice is indicated with a pipe symbol "|" (e.g.

[7] https://cloud.google.com/storage/docs/json_api/
[8] http://json-schema.org/

```
{
"gender": "male" | "female"
}
```

Arrays

When an array appears in a schema description, we only represent an example of the first element in the array, followed by an ellipsis ("..."). This means that the element may appear 0 or more times, unless otherwise specified in the description.

```
{
"name": <string>,
"phones": [
      {
"type": <string>,
"number": <string>
      },
      ...
  ]
}
```

11.2 API KEY

Clients will be required to provide an API key, in order to verify that they have the right to access the APIs. This API key shall be provided with the following HTTP header in every request:

```
Authorization: Bearer <API_KEY>
```

Where API_KEY is a 20-byte random secret value, encoded in BASE64. Each organisation accessing the API shall get a distinct API_KEY. If a client makes a request without specifying an API key, or with an unknown API key, the server will respond with HTTP error code 401 (Unauthorized) and provide the following HTTP header in the response:

```
WWW-Authenticate: Bearer realm="https://star.watch/api/v1/continuous"
```