Funded by EU's Horizon 2020

D2.1

CLOUD-BASED SERVER AND DATA ARCHITECTURES SPECIFICATION

Supporting Active Ageing through Multimodal coaching

DISCLAIMER

This document reflects the opinion of the authors only and not the opinion of the European Commission. The European Commission is not responsible for any use that may be made of the information it contains.

AKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No.769661.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

Supporting Active Ageing through Multimodal coaching

DELIVERABLE DOCUMENTATION SHEET

Deliverable:	*D 2.1*
WP №	*2*
Title:	*Cloud-based Server and Data Architectures Specification*
Author(s):	*Dimitar Skripkin, Dimo Kapnilov, Irina Stoyanova, Krasimir Kostadinov, Georgi Kushev, Rodislav Rusev*
Contributor(s):	*Bernard Ženko, Martin Žnidaršič*
Type:	*Report/Documentation*
Version:	*1.0*
Submission Due Date:	31.09.18
Dissemination level (CO/PU):	Public
Copyright:	*©SAAM Project - All rights reserved*

☒	Approved by the WP Leader
☒	Approved by the Technical/Exploitation Manager[1]
☒	Approved by the Coordinator
☒	Approved by the PSC

[1] Choose Technical Manager for Deliverables in WP1-7,10 and Exploitation Manager in WP 8-10

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

PUBLISHABLE SUMMARY

This Document serves to describe the cloud-based server Architecture of the SAAM system. It describes the components of the architecture and their interoperability as well the specific technology stack that will be used for the implementation of the system. This is an initial version of the document to be followed by a final version where changes based on the discovered necessities for the system outlined, during the pre-piloting and testing period of SAAM are taken into consideration. The last version of the document can be found HERE. In order to access the link you need to have a Teamwork account and be added to the SAAMH2020 Project.

The Document is primarily aimed at the software designers and developers within the SAAM project as well as at the future administrators of the System.

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 769661

QUALITY CONTROL ASSESSMENT SHEET

Version	Date	Comment	Name of author/reviewer/contributor
V0.1	*26.09.2018*	*First Draft*	*Dimitar Skripkin, Dimo Kapnilov, Krasimir Kostadinov, Georgi Kushev, Irina Stoyanova*
	5.10.2018	*Review First Draft*	*Aljaž Osojnik*
		Contributions	B*ernard Ženko, Martin Žnidaršič*
V0.2	*23.11.2018*	*Second Draft*	*Dimitar Skripkin, Dimo Kapnilov, Krasimir Kostadinov, Georgi Kushev, Irina Stoyanova, Rodislav Rusev*
	5.12.2018	*1st Peer review*[2]	*Mihael Mohorčič*
	6.12.2018	*2nd Peer review*	*Vera Veleva*
V1.0	*07.12.2018*	*Final Draft*	
	07.12.2018	*WP Leader approval*	
	10.12.2018	*Coordinator approval*	
		EAB Review	*N.a.*
	14.12.2018	*PSC approval*	
	19.12.2018	*Submission to EC*	

HISTORY OF CHANGES

For updating the Deliverable after submission to the EC if applicable

Version	Date	Change
V1.0		

[2] Each Deliverable to have one or two experts (partners) assigned for Peer review. The experts should be different from the Experts, contributing to its creation. List of Deliverables and Peer review experts to be decided at Kick-off.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

Supporting Active Ageing through Multimodal coaching

PROJECT DOCUMENTATION SHEET

Project Acronym:	*SAAM*
Project Full Title:	*Supporting Active Ageing through Multimodal coaching*
Grant Agreement:	*GA № 769661*
Call identifier:	*H2020-SC1-2017-CNECT-1*
Topic:	*Personalised coaching for well-being and care of people as they age*
Action:	*Research and Innovation Action*
Project Duration:	*36 months (1 October 2017 – 30 September 2020)*
Project Officer:	*Jose Valverde Albacete*
Coordinator:	*Balkan Institute for Labour and Social Policy (BILSP)*
Consortium partners:	*Jožef Stefan Institute (JSI)* *University of Edinburgh (UEDIN)* *Paris-Lodron Universitat Salzburg (PLUS)* *Scale Focus AD (SCALE)* *Interactive Wear AG (IAW)* *Univerzitetni rehabilitacijski inštitut Republike Slovenije (SOČA)* *Nacionalna Katolicheska Federacia CARITAS Bulgaria (CARITAS)* *Bulgarian Red Cross (BRC)* *Eurag Osterreich (EURAG)*
website:	*saam2020.eu*
social media:	*#saam2020, #saamproject*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

ABBREVIATIONS

SAAM	Supporting Active Ageing through Multimodal Coaching
ELK	Elasticsearch Logstash and Kibana
MQTT	Message Queuing Telemetry Transport
MQ	Message Que
API	Application programming interface
DB	Database
JSON	JavaScript Object Notation
AMQP	Advanced Message Queuing Protocol
STOMP	Streaming Text Oriented Messaging Protocol
OCI	Open Container Initiative
SSO	Single sign-on
OP	OpenID
TLS	Transport Layer Security
MSSQL	Microsoft SQL Server
REST	Representational State Transfer
VPN	Virtual Private Network

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 769661

CONTENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

TABLE OF FIGURES

ANNEXES

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

1. INTRODUCTION

The SAAM system architecture consists of a cloud-based server that will be used for data and user management, server-side software and client-side software, which will run locally on devices of each user. The purpose of the current document is to describe the Cloud-based Server portion of the system and the respective Data Architecture.

On a high-level the server side will be responsible for:

- Common mediation;
- Analytics and storage platform;
- SAAM platform management interface for SAAM administrators;
- Providing common services to SAAM clients (user sided components) like:
    - Virtual Coach services which require high data volume analysis or CPU intensive operations;
    - communication with other users from their social circles;
- Acting as a common storage for personal and behavioural data for SAAM Users;
- Support social circles functionality of SAAM system;
- Providing interfaces for integration with “in-house hardware” and third party (external vendors’ existing) services providers like medical services, volunteering and other kind of information services, smart city and weather conditions services, etc.

It is implemented as easy scalable cloud-based solution with operative transactional processing part with micro services for day to day operations and big data part of collecting and analysing historical data.

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 769661

2. HIGH-LEVEL ARCHITECTURE OVERVIEW

The SAAM Cloud-based Server Architecture can be logically divided in 7 sections, described in more details in the current document:

- External Communications Point
- Data Layer
- Containerization
- ELK
- Microservices
- In-memory caching
- Security

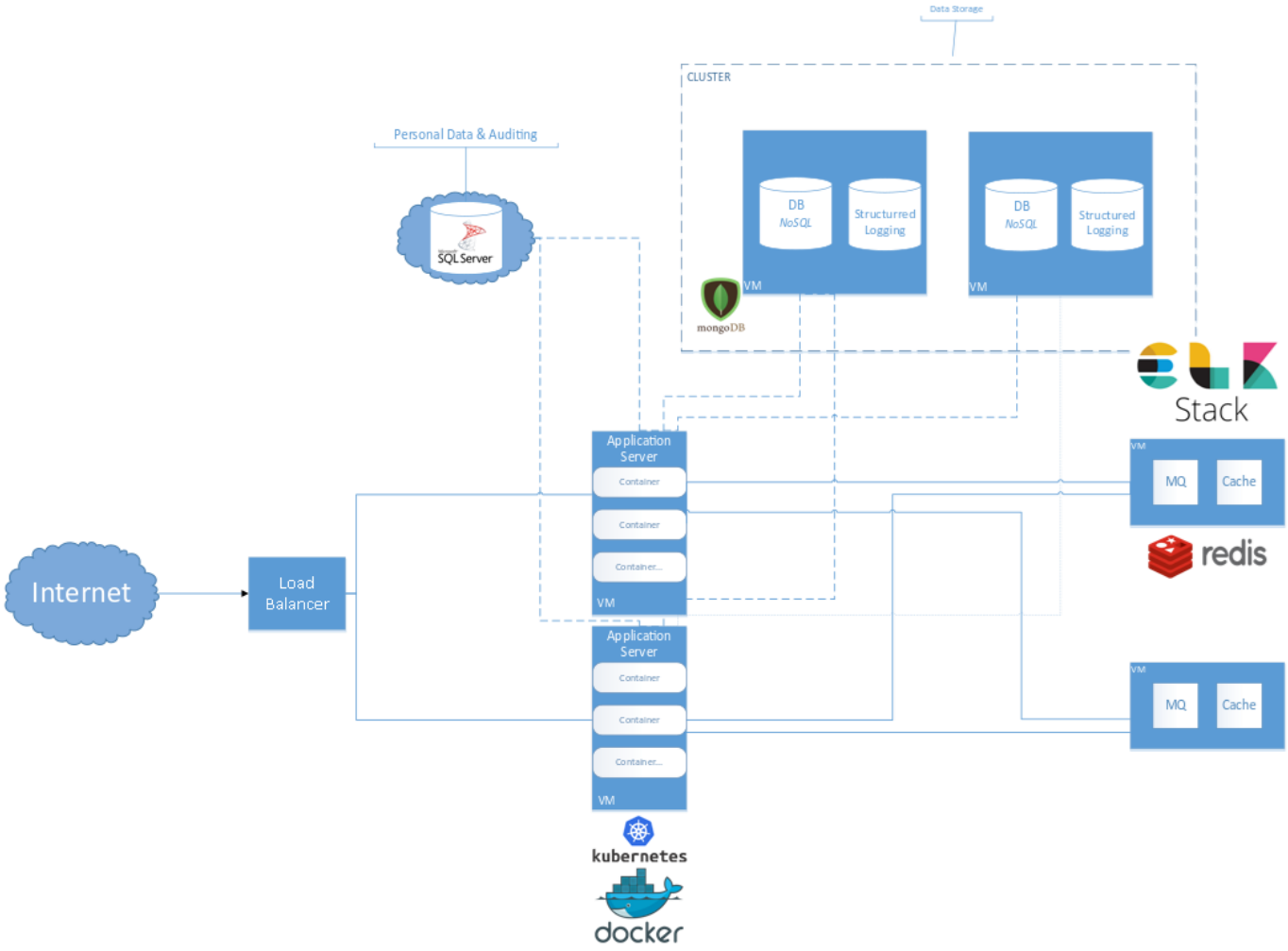

Figure 1 - High-level Architecture

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

The processes of interaction between those sections may vary depending on the type of event that is triggering them, e.g. the Gateway is sending data to the Server vs. a User wanting to change some information on their user profile:

The Gateway sends data to the Server

1) The SAAM Gateway has sent data to the Cloud-based server;
2) The Data is received by the RabbitMQ via MQTT protocol;
3) A dedicated message listener will process the Data and then store it in the Mongo DB;
4) The same data (if necessary) will trigger a coaching action;
5) If coaching action is triggered – the event will be logged in ELK;
6) We are requesting user information from Reddis (if the data is available there – if not a request is made to the Azure DB, which is then stored in Reddis) based on the location ID provide in the received device data;
7) The user will receive the relevant to the coaching action notification on their mobile device or Web Portal via SignalR Connection.

User wants to change her Profile photo:

1) First she needs to log-in SAAM via the Web portal or Mobile App;
2) The data generated by this event is being distributed via the Load Balancer to the least loaded instance of the Authentication API;
3) The request to the Microservice is logged in ELK;
4) A check is performed to insure that the user's credentials belong to an existing user in the Azure DB.
5) If the credentials are accurate a search for the user's profile information in Reddis will be triggered.
    - If the Information is available in Reddis a token will be generated and sent to the user as a response
    - If the information is not available in Reddis, a request to Azure will be made and information will be provided to Reddis, after which a token will be generated and sent to the user as a response.
6) Now the User is successfully logged-in the system. She goes to the Edit User Profile and uploads new photo.
7) The data generated by this event is being distributed via the Load Balancer to the least loaded instance of the User API;
8) The request to the Microservice is logged in ELK;
9) The new profile picture is sent to Azure and the Reddis cache related to the user is invalidated;
10) Since the solution is response based, if the process has been completed successfully, an https success response will be sent

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

3. EXTERNAL COMMUNICATION POINTS

Currently the System will be communicating with external entities via two external communication points – Message Broker and Load Balancer

a. RabbitMQ Master Instances

When you deploy the RabbitMQ (min 3 instances for a cluster) –you have one "master" instance with two "slave" instances. The Master instance receives all messages and distributes them to the other two instances. The Master instance in our case will be accessed via the MQTT Protocol.

b. Load Balancer

The load balancer distributes https calls to the SAAM API, between application Microservice instances. The distribution process depends predominantly on the load of those instances.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

Figure 2 - High-level Architecture Overview

4. DATA LAYER

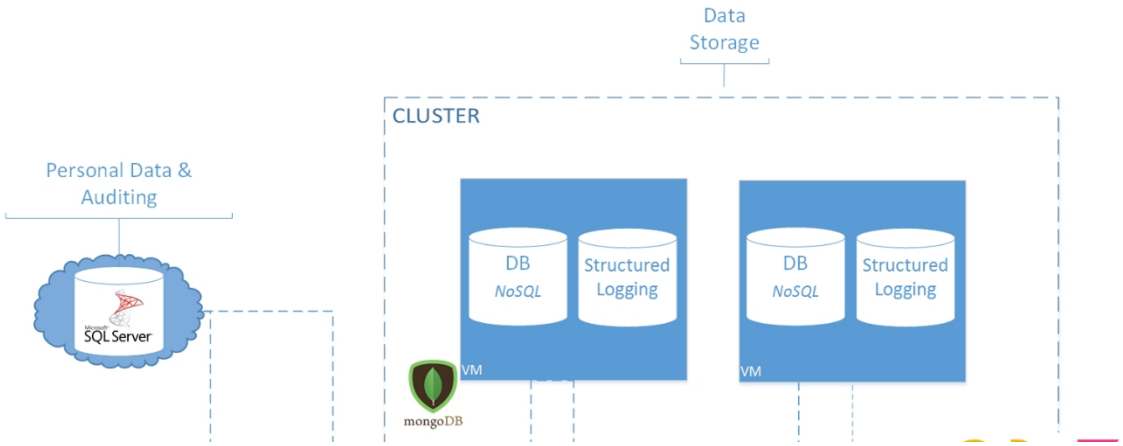

Figure 3: Data layer

A combination of separated relational and document databases will be used for storing sensitive user data, and sensor related data (either raw or pre-processed) acquired at users homes. Maintaining separate databases will result in additional security to the system since the physical instances will be located in a different places – the Data storage Cluster will be located in a data centre in Bulgaria, while the Personal Data will be stored in Azure – EU (Location to be determined upon contract signing with the provider). The connection between the different data storage entities relies on a mapping between the location ID provided by Edge Gateway device and the relevant user stored in Azure.

4.1. In-memory caching for user data

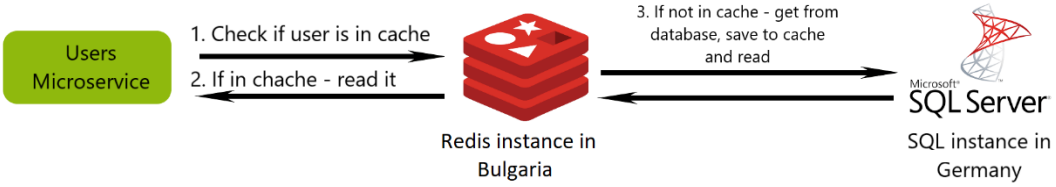

Figure 4 In-memory caching for user data

In order to overcome any performance problems due to the separate databases' locations, in-memory caching for user data will be used via Redis. When a request for the user data is made for example to instance located in Bulgaria the system will first check if there is user data kept locally in-memory and it

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

will return it. If there is no cached data the system will get it from the long-term cloud storage and save to Redis for further use.

4.2. Data Storage

The data is stored mainly in three databases:

- MSSQL - Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).
- MongoDB - Free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemata.
- Redis (Remote Dictionary Server) is an open-source in-memory database project implementing a distributed, in-memory key-value store with optional durability.
- Elasticsearch is a NoSQL DB, based on Apache Lucene and this is where we will be storing the SAAM system logs.

4.3. Logical Data Organization

All the data collected at the server will be available via Views that allow for querying by *location_id, source_id* and *time stamp intervals.* The *source_id* will uniquely identify which quantity is measured (from hardware sensors) or constructed (during feature construction). The preliminary set of needed Views is:

- For measurements the data will be organized according to the devices that generate them:
    - smartphone
        - MicroHub bracelet
        - MicroHub wearable
        - MicroHub bed sensor
        - PMC Device
        - UWB
        - ambient sensor
        - standard AV sensor
        - advanced AV sensor
        - Savvy ECG sensor
        - foodscale
        - BLE beacon

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 769661

- For features the data will be organized according to the coaching domains (see the D1.4 on use cases):
  - mobility
  - activity
  - sleep
  - social
  - cardiovascular
  - cognitive
  - nutrition

Using such pre-specified queries will optimise the data retrieval processes needed for data analysis and modelling, since each query will mostly be specified by the view name, filtering out all data known to be irrelevant for the given domain. The contents of each of these views will be specified as a list of *source_id* values, i.e., for each view we will have a list of measurements and features that are present within the view. The specification of these lists, especially the ones for the features, will change during the development of the system, depending on the data analysis results.

4.4. Data Back-up

4.4.1. Azure

The SAAM system will depend on Azure SQL Database. Azure SQL Database is implementation of the latest stable MSSQL server which is configured and optimised for cloud usage, with out of the box resiliency policies and procedures. The Azure SQL Database supports automated backups and 'Point in Time Restore' which enables us to restore the entire database to some point in time within the last 35 days.

4.4.2. Cloud-based Server

The SAAM system infrastructure will have incremental backups of the VMs which are done once a day. Incremental backup is type of backup that only copies part of the data that is changed after successful backup. This will allow us to have relevant backup with minimal system load.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

5. MICRO SERVICES

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities. Idea behind this approach is to allow easier scalability of part of the system when they are under heavier load than others. For example in the SAAM system one microservice would be the SensorAPI microservice which will be responsible for adding and getting sensor data. Another one UserAPI which is responsible for adding, changing or deleting a user. Whit this approach if we have a lot of devices pushing data to the SAAM system we can have 3 instances of SensorAPI and 1 of UserAPI. More detailed information about the micro services in SAAM system can be found in Deliverable 2.9 Hardware and Software APIs and Interfaces.

6. PLATFORM SCALABILITY

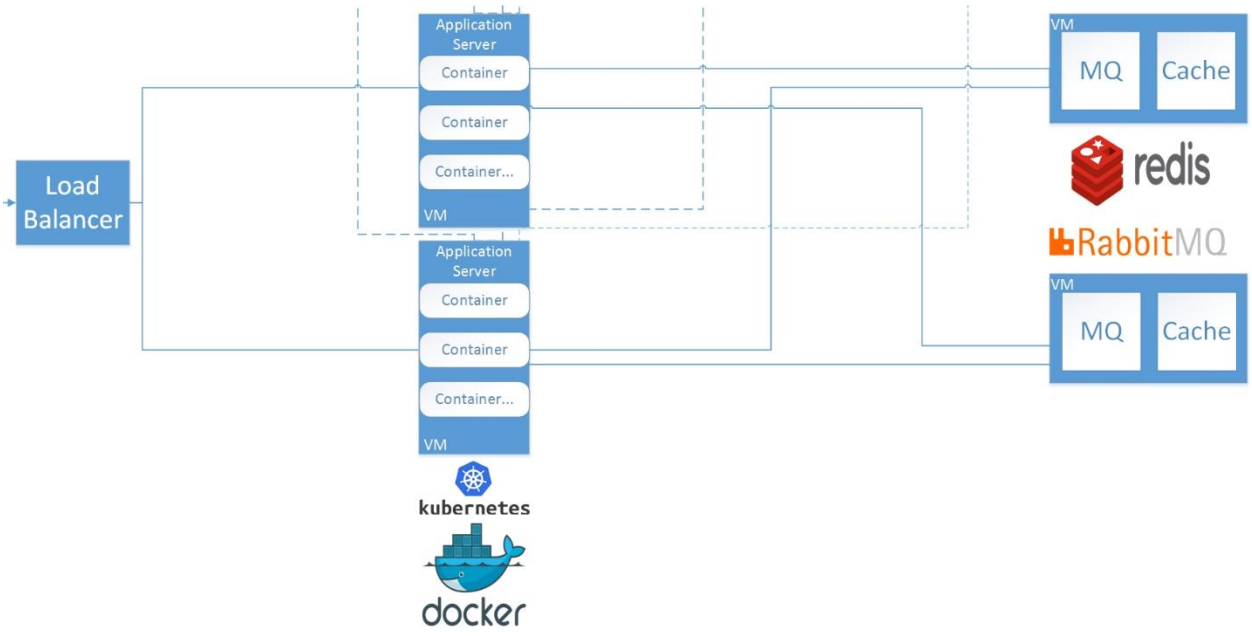

Figure 5: Platform scalability

When the system gets a request, it'll be routed through load balancer which will choose the least loaded instance. When it is necessary the containers will communicate through the message queue to minimize the coupling between different system components. For the SAAM system the message queue will be RabbitMQ.

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 769661

RabbitMQ is an open source message broker software (sometimes called message-oriented middleware) that originally implemented the Advanced Message Queuing Protocol (AMQP) and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol (STOMP), Message Queuing Telemetry Transport (MQTT), and other protocols.

7. CONTAINERIZATION

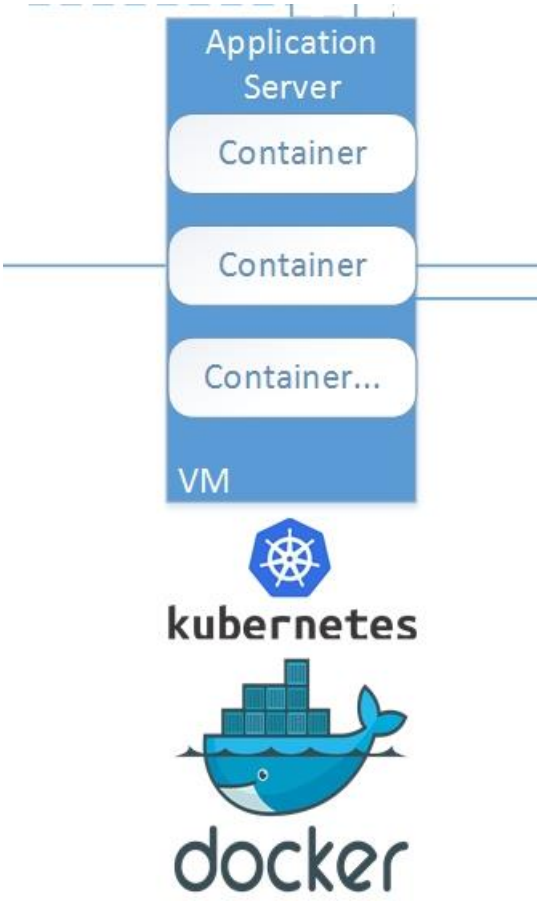

Figure 6: SAAM Containerization

The approach applied in SAAM is to package the different micro services that constitute an application into separate containers, and to deploy those containers across a cluster of virtual machines. That

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

implementation requires container orchestration - a tool that automates the deployment, management, scaling, networking, and availability of container-based applications.

Docker is a computer program that performs operating-system-level virtualization, also known as containerization. It is used to run software packages called containers. In a typical example use case, one container runs a web server and web application, while a second container runs a database server that is used by the web application. Containers are isolated from each other and bundle their own tools, libraries and configuration files, though they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and are thus more lightweight than virtual machines. Containers are created from "images" that specify their precise contents. Images are often created by combining and modifying standard images downloaded from repositories.

For container orchestration SAAM system will depend on the open-source orchestration system called Kubernetes. Kubernetes automates the process of deploying and managing multi-container applications at scale. While it works mainly with Docker, it can also work with any container system that conforms to the Open Container Initiative (OCI) standards for container image formats and runtimes. (The Kubernetes Installation Steps, are described in ANNEX I of this document)

8. LOGGING AND AUDITING

Figure 7: ELK stack

By combining Elasticsearch, Logstash, and Kibana (ELK/Elastic Stack), Elastic is an end-to-end stack that delivers actionable insights in real time from almost any type of structured and unstructured data. Together, these different open source products are most commonly used for centralized logging in IT environments (though there are many more use cases for the ELK Stack including business intelligence, security and compliance, and web analytics). Logstash collects and parses logs, and then Elasticsearch indexes and stores the information. Kibana then presents the data in visualizations that provide actionable insights into one's environment. For auditing of the SAAM system the information necessary for handling system issues (system errors, event logs, etc), will be directly redirected to the ELK

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 769661

Kubernetes. The received data will be processed and later on visualized for easier analysis. Kibana will be used to visualize the usage of the virtual machines hosting the containerisation (errors, users logged in, etc.).

For example in the SAAM system if request to change the user address has been made the following will happen.

1. Request data is logged in the console output;
2. Kubernetes will redirect the console output to ELK;
3. Elasticsearch will save the data in indexed structure that can be queried later

9. PLATFORM SECURITY (SERVER-SIDE)

9.1. Tokenization

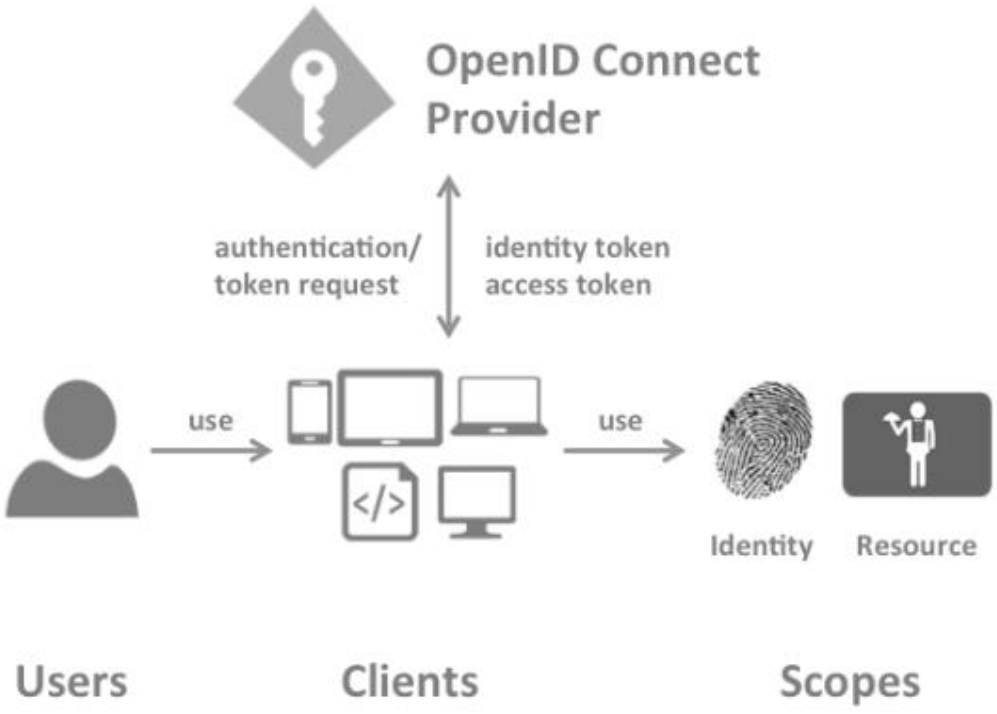

Figure 8: User Access Platform Security

The whole system will use single sign-on in order to have single account for the whole system and improve the user experience.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

When a User logs in SAAM a token request is made, with a certain scope (e.g. profile, userapi, sensorapi, etc). A scope is a set of predefined Claims (e.g. name, family_name, picture, gender, etc.) that are used to request access to different resources. If the inputted credentials exist in the DB an identity token, access token or both will be issued.

9.1.1. Token lifespan

Since access tokens have finite lifetimes, refresh tokens allow requesting new access tokens without user interaction. SAAM will support refresh tokens with a sliding lifetime of a refresh token, with the exact number of days/second being decided later on in the project.

9.1.2. Token Creation

JSON Web Token is a JSON-based open standard (RFC 7519) for creating access tokens that assert some number of claims. The tokens are signed by one party's private key (usually the server's), so that both parties (the other already being, by some suitable and trustworthy means, in possession of the corresponding public key) are able to verify that the token is legitimate. The tokens are designed to be compact, URL-safe, and usable especially in web browser single sign-on (SSO) context.

Clients request tokens from the OP. Depending on the scopes requested, the OP will return an identity token, an access token, or both. An identity token represents the outcome of an authentication process. It contains at a bare minimum an identifier for the user (called the sub aka subject claim). It can contain additional information about the user and details on how the user authenticated at the OP. An access token allows access to a resource. Clients request access tokens and forward them to an API. Access tokens contain information about the client and the user (if present). APIs use that information to authorize access to their data.

9.1.3. Token Properties

A server could generate a token that has the claim "logged in as admin" and provide that to a client. The client could then use that token to prove that it is logged in as admin. JWT claims can be typically used to pass identity of authenticated users between an identity provider and a service provider, or any other type of claims as required by business processes.

9.1.4. Issues Token logs

A log with the issued tokens will be supported for SAAM with the following attributes:

- Issued date

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 769661

- Issued Scopes
- Issued to User
- Lifespan (optional)

9.2. **Data Encryption**

The system will use TLS (Transport Layer Security) protocol which aims primarily to provide privacy and data integrity between two or more communicating computer applications. When secured by TLS, connections between a client (e.g., a web browser/mobile application) and a server (e.g., SAAM domain) have one or more of the following properties:

- The connection is private (or secure) because symmetric cryptography is used to encrypt the data transmitted. The keys for this symmetric encryption are generated uniquely for each connection and are based on a shared secret that was negotiated at the start of the session. The server and client negotiate the details of which encryption algorithm and cryptographic keys to use before the first byte of data is transmitted. The negotiation of a shared secret is both secure (the negotiated secret is unavailable to eavesdroppers and cannot be obtained, even by an attacker who places themselves in the middle of the connection) and reliable (no attacker can modify the communications during the negotiation without being detected).
- The identity of the communicating parties can be authenticated using public-key cryptography. This authentication can be made optional, but is generally required for at least one of the parties (typically the server).
- The connection is reliable because each message transmitted includes a message integrity check using a message authentication code to prevent undetected loss or alteration of the data during transmission.

In addition to the properties above, careful configuration of TLS can provide additional privacy-related properties such as forward secrecy, ensuring that any future disclosure of encryption keys cannot be used to decrypt any TLS communications recorded in the past.

In order to secure further the integrity and privacy of the stored data the System will be constructed in a manner preventing any external access to the Platform's DBs. The Azure MSSQL instance will be accessible only from a specified IP with the help of VPN when necessary and the data stored in Mongo DB will be accessible solely via micro service instances.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

10. TECHNOLOGY STACK

The server side system will be implemented using a set of proven and reliable technologies:

- MSSQL (in Azure)
- MongoDB (on premises - sensor data and logs)
- ELK Stack
    - Elasticsearch
      Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data so you can discover the expected and uncover the unexpected.
    - Logstash
      Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it.
    - Kibana
      Kibana lets you visualize your Elasticsearch data and navigate the Elastic Stack, so you can do anything from learning why you're getting paged at 2:00 a.m. to understanding the impact rain might have on your quarterly numbers.
- RabbitMQ
- Redis
- Kubernetes
- Docker
- WebRTC
- ASP .Net Core 2

For data analysis we will use the following technologies that will be available within Docker containers:

- Python with additional libraries (pip management system)
- Java with Maven
- Octave
- R

The specific technologies visualised in the figures in Section 1.1 as well as the ones listed here could be replaced in the process of development if necessary.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

11. HARDWARE COMPONENTS

The server side system will run on a set of hardware components with the following technical specifications.

11.1. A High Power Rack Mountable Server with a Redundant Power Supply

- 2U dual-socket server
- Support two processor sockets with up to 22-core's each, up to 55 MB of L3 cache, up to 2400 MHz memory speeds and up to 9.6 GT/s QPI interconnect links
- Up to 1.5 TB of memory capacity with 64 GB load-reduced DIMMs, or LRDIMMs
- Supplied with 2 CPU processors with 10 cores each, 2.4GHz
- Supplied with 512 GB RAM, 2400MHz
- Supplied with 2 x 300 GB SAS Disk, 12GPBS
- Supplied with FC HBA, 16GB and FC cables to connect it with external storage
- Supplied with 2 power supplies
- All of the cables required to power up the server
- 3 years' warranty

11.2. Rack Mountable External Storage

- Scalable, modular storage with dual-active intelligent array node canisters
- Host connectivity which supports FCoE, 16 Gb FC
- 12 Gb SAS drive-side connectivity with support for 12x 3.5-inch large form factor (LFF) or 24x 2.5- inch small form factor (SFF) drives in the control enclosure
- Support of high-performance SAS SSDs, performance-optimized enterprise SAS HDDs, or capacity-optimized enterprise NL SAS HDDs;
- Standard functions including virtualized internal storage, thin provisioning, one-way data migration, FlashCopy snapshots and embedded GUI.
- Dual-port drives with automatic drive failure detection and RAID rebuild with global hot spares
- Redundant hardware, including power supplies and fans
- Hot-swappable and customer replaceable components
- Automated path failover support for the data path between the server and the drives
- All of the cables required to power up storage
- 12 x 1.2TB 2.5" 10K HDD

Or Rack Mountable Storage expansion compatible with Lenovo Storage V3700 V2 with following technical specification:

- Redundant power supplies

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

- 12 x 1.2TB 2.5" 10K HDD
- Interface SAS 12Gb/s
- Supports up to 24 2.5" HDD
- All of the cables required to power up the expansion

11.3. Security Appliance

- Provides Firewall, NAT, IPSec, Routing, MPLS and Switching capabilities
- 1U rack mountable
- Capacity of 5 gigabits per second (Gbps)
- Has eight 1 G Ethernet ports, eight 1 G SFP ports, one management port,
- Has 4 GB of DRAM memory, 8 GB of flash memory
- Firewall support with key features such as VPN
- Intrusion Detection and Prevention (IDP), AppSecure, and UTM
- QoS
- Next Day Support for hardware replacement

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No 769661

ANNEX I – BASIC SYSTEM SET-UP

Minimal requirements for one instance - VM:

HDD – 50G

RAM – 8 GB

CPU Cores – 4

OS – CentOS Linux

Environments:
Test

Dev

Prod

Kubernetes Installation Steps

On Master Kuberenetes Node:

Root user

1. hostnamectl set-hostname kube-master

2. vim /etc/selinux/config --- Disable selinux

3. setenforce 0

4. systemctl stop firewalld.service && systemctl disable firewalld.service

5. yum -y install yum-utils

6. yum-config-manager --add-repo https://yum.dockerproject.org/repo/main/centos/7

7. yum -y update

8. yum -y --nogpgcheck install docker-engine-1.12.6-1.el7.centos.x86_64

9. systemctl enable docker && systemctl start docker

10. cat <<EOF > /etc/yum.repos.d/kubernetes.repo

> [kubernetes]

> name=Kubernetes

> baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-$basearch

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

> enabled=1

> gpgcheck=1

> repo_gpgcheck=1

> gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

> EOF

11. yum install kubelet kubeadm kubectl -y

12. cat <<EOF > /etc/sysctl.d/k8s.conf

> net.bridge.bridge-nf-call-ip6tables = 1

> net.bridge.bridge-nf-call-iptables = 1

> EOF

13. sysctl --system

14. mkdir -p /etc/cni/net.d

15. chmod -R 777 /etc/cni

16. kubeadm init --pod-network-cidr=10.0.4.0/24 --apiserver-advertise-address=10.0.4.77

17. mkdir -p /home/kubeuser/.kube

18. cp -i /etc/kubernetes/admin.conf /home/kubeuser/.kube/config

19. chown -R kubeuser:kubeuser /home/kubeuser/.kube

###Regular user (kubeuser)

1. kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

2. kubectl get nodes

On Slave`s Kuberenetes Node:

ROOT user

1. yum install docker

2. systemctl enable docker && systemctl start docker

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661

3.cat <<EOF > /etc/yum.repos.d/kubernetes.repo

> [kubernetes]

> name=Kubernetes

> baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-$basearch

> enabled=1

> gpgcheck=1

> repo_gpgcheck=1

> gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

> EOF

4. yum install kubelet kubeadm kubectl

5. cat <<EOF > /etc/sysctl.d/k8s.conf

> net.bridge.bridge-nf-call-ip6tables = 1

> net.bridge.bridge-nf-call-iptables = 1

> EOF

6. mkdir /etc/cni/net.d

7. chmod -R 777 /etc/cni/net.d

8. check gorup on kubernetes in file: /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

---

Generate new token for join a node:

[root@master] $ kubeadm token create --print-join-command

list all tokens:

[root@master] $ kubeadm token list

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 769661