Master of Science in Telecommunication Systems
October 2020

Investigation and Implementation of Federation Mechanisms of SVP

Babar Khan

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Telecommunication Systems. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Babar Khan
E-mail: bakh17@student.bth.se

University advisor:
Kurt Tutschku, Prof. Dr.
Department of Computer Science and Engineering (DIDA)

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

The development of AI application on the edge devices require integrated data, algorithms, and tools. Big companies like Google and Apple have integrated data, algorithms, and tools for building end to end systems with optimized and dedicated hardware for deep learning applications. The Bonseyes [2] EU H2020 collaborative project is an open and expandable AI platform. Bonsey's provides access to advanced tools and services that can be obtained from the data market place and eco-system of collaborative leading academic and industrial partners for adding AI to embedded products. Bonseyes AI Marketplace proposed an end to end AI pipeline to overcome these requirements for the development and deployment of AI solutions on edge devices. In Bonseyes, a Secure virtual premise (SVP) is a secure and protected area for the collaborative and systematic development of AI using Machine Learning AI Pipelines. The centralized SVP has limitations like scalability, reliability, and load balancing. This thesis work focuses on the enhancement of SVP and its mechanisms to adopt a distributed and federated architecture for better performance and fast development of AI applications. It investigates various federation mechanisms and implements a distributed and federated SVP. A Marketplace rendezvous host is designed and implemented from where multiple distributed compute resources can be started, controlled, and stopped by a user. Users and distributed locations related data are stored in an SQLite relational database. Communication between entities of distributed and federated SVP is enabled using HTTPs protocol and PKI Infrastructure is used for authentication and authorization of users. We evaluate the performance of our implementation by calculating the start-up time of multiple resources until a user can perform AI Engineering. The results show that time is directly proportional to the number of nodes started.

Keywords: Bonseyes Marketplace, AI Engineering, Resource Federation, Federation mechanisms, Distributed Secure Virtual Premise.

ACKNOWLEDGMENT

My sincere gratitude goes to my advisor Prof. Dr. Kurt Tutschku for his patience and invaluable guidance. Without his support, it would not have been possible to accomplish this thesis. His feedbacks and research experience helped me in critically viewing key topics and writing this thesis.

My heartful gratitude goes to my wonderful family and friends. Their constant prayer, support, and motivation have been a great source of energy throughout my years of study.

CONTENTS

List of Figures

List of Tables

1 INTRODUCTION

Even with the emergence of promising AI technologies like Deep Learning in recent years, the development of an artificial intelligence system (AI) is challenging. This needs the creation and cure of broad data sets, the search for suitable hyperparameters for the AI model, the use of suitable training algorithms, and benchmarking to determine whether the AI program developed fulfills the functional and quality criteria of the challenge to be addressed. Current AI development methods rely on experimentation by professional data engineers and scientists, which require enough time and resources to produce AI of sufficient maturity. More work is being conducted to facilitate the development of an AI program with many approaches being followed in the AI pipelines: improved data set preparation techniques and software, improved model architectures, improved hyperparameter optimization techniques and tools, and improved hardware for more effective AI inference. Bonseyes [1] pursues an approach of AI innovation that is beneficial to market innovators of minimal AI experience and can be seen as a supplement to AI pipeline-based strategies facilitating cooperation with experienced AI firms.

The Bonseyes [2] EU H2020 collaborative project is an open and expandable AI platform. Bonsey's provides access to advanced tools and services that can be obtained from the data market place and eco-system of collaborative leading academic and industrial partners for adding AI to embedded products. This will reduce the development time and cost by reusing data. In Bonseyes, the secure virtual premise (SVP) [19] is a protected, distributed, and decentralized space where the collaborators can securely integrate AI artifacts and tools into a shared AI pipeline. It is not a physical area nor it is a single dedicated physical compute infrastructure but is an emulated space comprising interconnected and virtualized computing and storage resources. Each artifact is contained in a docker container and shielded by a Bonseyes layer (BL). The Docker image provides a complete runtime environment in addition to the kernel, ensuring consistency in the application runtime environment. It can achieve a second or even millisecond startup time, which greatly saves development, testing, and deployment time, avoids common servers, and resources are easily affected by other users. Each artifact and tool is associated with the end-user provided license that defines the term and condition of its use. The SVP provides the isolation of AI artifacts and tools executed inside the

AI pipeline from the end-users and enforces and checks the applicable licenses for each user interaction.

This thesis aims to improve the SVP and its mechanism to adopt a distributed architecture for AI engineering. Currently, we have a single SVP location where AI artifact owners and AI artifact users are collaborating securely. However, in practice, artifact owners may want to deploy the artifacts in different locations to balance the load and make their solution scalable and flexible. Subsequently, an artifact user may want to use artifacts distributed across the different locations to execute there AI pipeline [3]. To solve the above-mentioned problems, our goal is to implement the federation of multi-location SVP. This will result in load balancing, high performance, scalability, and flexibility. Figure 1 depicts the current centralized SVP and the future SVP with distributed resources. In current SVP we have a single location, user, and artifact for AI engineering whereas in future distributed and federated SVP we want to enable multiple locations, more users, and multiple artifacts for trusted and secure collaborative AI development.

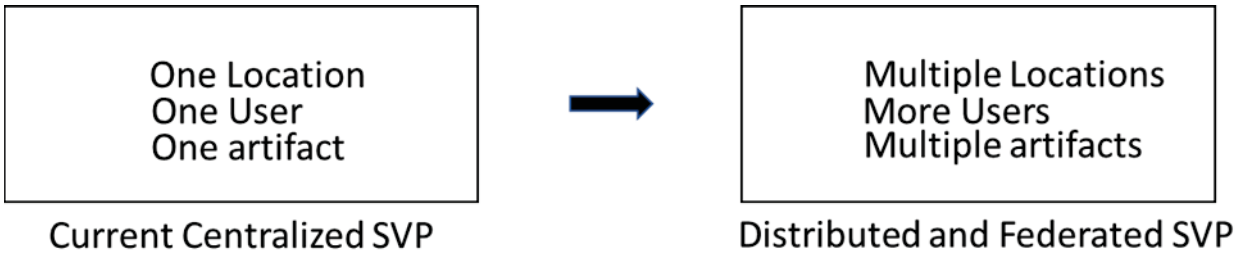

Figure 1 - From current Centralized SVP to Distributed and Federated SVP

1.1 Problem Statement/Motivation

The current centralized SVP has limitations like scalability, reliability, and load balancing. For the better performance and utilization of the collaboration model of the SVP, the distributed remote resources must be federated inside the SVP. These individual resources contain microservices that are AI artifacts or tools. Technologies like Openstack [4] and Kubernetes [5] have the services capable of federating distributed resources. But here, we are considering a more simplified approach with minimal software dependencies for user ease. Therefore, we require an automated federation mechanism to federate different remote secure locations. The

automated federation will result in a trusted, secure, enhance computation, and better user experience.

1.2 Aims and Objectives

The aim of the thesis is to investage and implement the federation of resources in SVP and to develop a federation concept. The thesis then aim to develop a GUI for the federation and to demonstrate and analyze the factional capability of the implementation.

The objectives of the thesis are:

- Investigate how an implementation of the SVP where locations, users, and artifacts can be added, can be achieved.
- Provide and demonstrate a program that starts, control, and stops an SVP with multiple locations, Users, and artifacts.
- Define data structures for controlling the SVP with multiple users and locations.
- Provide and validate an implementation for the secure execution of remote multiple locations.

1.3 Research Questions

RQ1: Which functions does one need to design and implement the federation of multi location SVP?

- How can user friendly configuration GUI for federated AI engineering be designed and implemented?
- Which information need to be displayed in the GUI for distributed AI engineering?
- Which functions are needed to start, control, and stop multi-location SVP?

RQ2: How the data need to be structured for setting up multiple distributed locations?

RQ3: How can these remote multiple locations securely be started-up and managed for the AI engineering task?

1.4 Main Contribution of this Thesis

In the BTH demonstrator of Bonseyes system, our contribution is to develop the federation concept by providing the implementation of the distributed SVP where locations, users, and owners can be added. Furthermore, to provide a program that starts, control, and stops an SVP with multiple locations, Users, and artifacts. Defining data structures for controlling the SVP with multiple users and locations and developing a GUI for the distributed resources of SVP.

1.5 Thesis Outline

The rest of the thesis is structured as follows:

1. Chapter 2 – Background: The concept of the Bonseyes Marketplace project and Secure Virtual Premise (SVP) is explained.
2. Chapter 3 – Related work: Work done related to the thesis is explained.
3. Chapter 4 – Method and technologies: Explains the method that we choose and the basic technologies (Microsoft Azure, Docker Containers, and Certificate Management) involved in the thesis.
4. Chapter 5 – Solution and Analysis: Explains the solution and analysis.
5. Chapter 6 – Answers to Research Questions: Research questions are answered.
6. Conclusion and future work: Includes conclusion and the future works of the thesis.

2 CONCEPTS OF THE BONSEYES MARKETPLACE AND ITS SECURE VIRTUAL PREMISE (SVP)

Engineering AI by small and medium sized enterprises needs increasingly specialization and collaboration to speed up the development and to increase its cost-efficiency. The Bonseyes aims to address these objectives by transforming the development of AI from a cloud-centric model to a collaborative edge device-centric model. Bonseyes provides access to advanced tools and services that can be obtained from the data market place and eco-system of collaborative leading academic and industrial partners for adding AI to embedded products. However, Collaboration raises trust and security problems. To address these issues, Bonseyes includes a Secure Virtual Premise (SVP), a distributed, decentralized space in which collaborators can securely integrate AI artifacts and tools into a shared AI pipeline. Each artifact and tool is associated with an end user-provided license that defines the terms and conditions of its use [19].

Bonseyes implements the collaborative AI engineering in the Cloud by offering an AI marketplace (MP), which is an open platform for trading AI artifacts, and the Secure Virtual Premise (SVP), which is a closed PaaS for distributed, secure, and trusted execution of AI pipelines. The MP aims to support the exchange of AI data and algorithms with third parties, under controlled forms dictated by licenses accompanying the AI artifacts. In this approach, an application developer can obtain licensed access (e.g., by paying fees) to sophisticated AI objects. The developer than can use the algorithms under the conditions covered by the licenses in the SVP to a) train models on data and b) validate the trained models. The access to these AI objects enables the developer to focus on the application, thus accelerating the development [19].

The concept of SVP is to specify a protected area where stakeholders can meet and collaborate successfully. However, the protected area here is not a single dedicated area or physical compute infrastructure but is an emulated space consisting of interconnected and virtualized compute and storage resources. These resources can be positioned at some random physical places or locations owned by different stakeholders. Thus it enables the use of cloud as well as edge devices like raspberry pi which are not computationally so strong. SVP is initiated on-demand only when it is needed with a high degree of automation [19].

2.1 Collaborative AI Engineering

Collaboration enables fast development of systems of Artificial intelligence by building on the expertise and results of companies which are specialized in AI. Such companies are working for years in the field of AI which can bring high-quality data, expertise, and performance models. These important factors can speed up the development of AI systems with good results. One of the main objectives of Bonseyes system is to support the collaboration of AI development in the secure virtual premise (SVP). A secure virtual premise with the distributed architecture would enable the collaboration of multiple innovators and developers. This will result in load balancing and high performance. Figure 2 below illustrates collaboration for the development of an AI-based app. The innovator company specifies the AI app requirements that monitor the state of the driver. AI developers based on their experience build an app believed to fulfill the requirements. Innovator tests the app with its secret data to verify if it fulfills the requirement or not. This cycle continues until it provides good results and leads the product to the market [19].

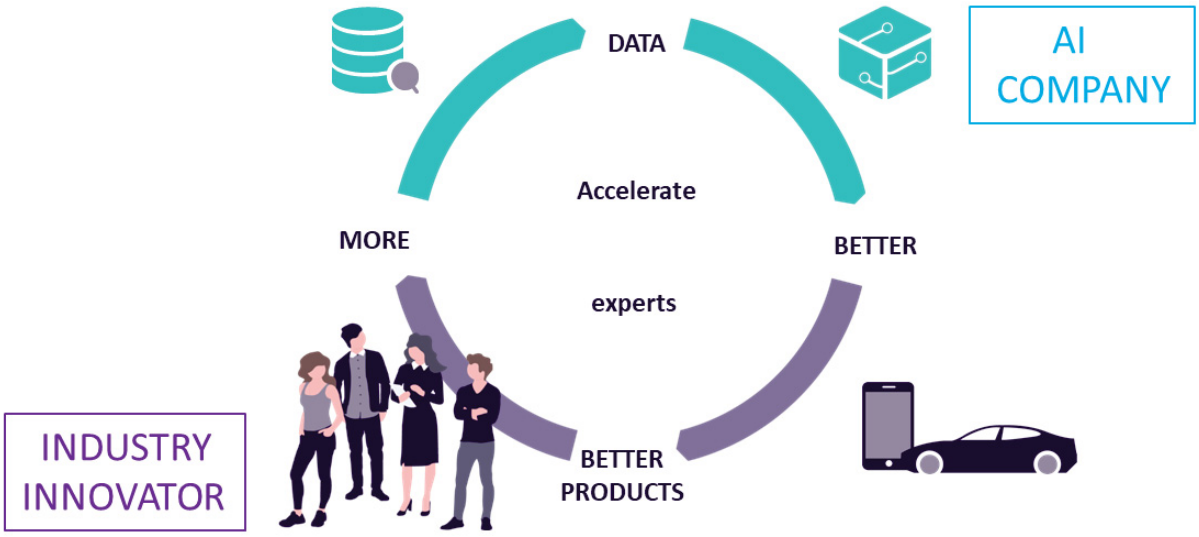

Figure 2 - Collaboration allowing industry innovators to leverage data and expertise of AI companies [19]

Collaboration gives a boost to AI development but indirectly it exposes the business secrets and results in loss of control over artifacts, such as data, models, or tools that must be shared.

Undisclosed agreements and business contracts offer a basis for clarification of the terms and conditions of collaboration. However, they do not allow automated enforcement of these terms and conditions in a way that guarantees that a third party will not misuse the trust.

2.1.1 AI Pipelines

The development of custom AI applications on edge devices requires integrated data, algorithms, and tools to train and deploy deep learning algorithms in a resource-constrained environment. This integration requires a high degree of skill in addressing both software and hardware requirements as well as the fragmentation of AI technologies, e.g., large and dispersed testing computation, reduced and limited inference capabilities, and portability difficulties between frameworks. These requirements are hurdles for small and medium companies who want to deploy AI solutions on edge devices. Big companies like Google and Apple have integrated data, algorithms, and tools for building end to end systems with optimized and dedicated hardware for deep learning applications. Bonseyes proposed an end to end AI pipeline to overcome these requirements for the development and deployment of AI solutions on edge devices. It provides integrated data, algorithms, and deployment tools together with the interconnecting individual task for an end to end development of AI products for embedded devices. The key benefits of such end to end development are scalability, flexibility, and reusability. Bonseyes end to end AI pipeline consists of four main components, see figure 3: data ingestion, Training, development optimization, and IoT hub integration [6].

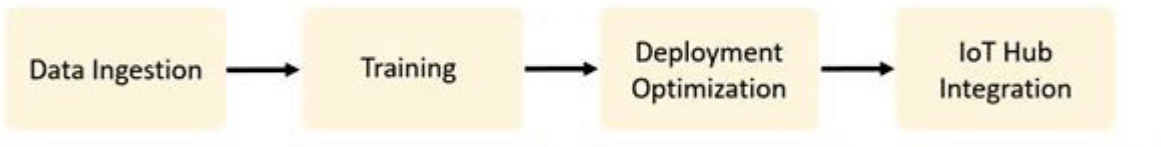

Figure 3 - AI pipeline interconnecting the four main individual tasks of end-to-end development [6]

In Bonseye system a major concept is the collaborative and systematic development of data-driven AI and Machine learning (ML) solutions. The systematic engineering is assisted by machine learning pipelines named as AI pipelines in the system. These pipelines automate the machine learning workflows and consist of modular steps to train a model or to create an AI application. Each step is considered a service and is regarded as an AI artifact together with

data objects. Every AI artifact is implemented in a docker container. Figure 4 shows the structure of two AI Pipelines in Bonseyes.

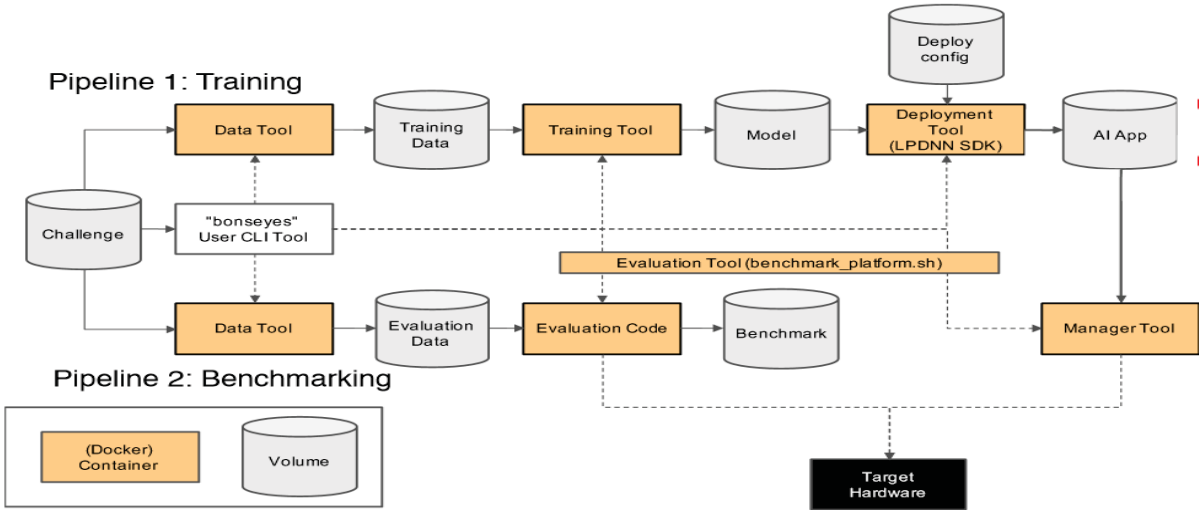

Figure 4 - Bonseyes Pipelines [19]

The first pipeline is a training pipeline consisting of a method for data extraction, a training tool, and a tool for deployment. The second pipeline is a Benchmarking Pipeline with a Data Extraction Method, an Evaluation Tool, and a Manager Tool. In the AI engineering phase of Bonseyes, the data is exchanged through file objects, e.g. volumes, between artifacts. Figure 4 Displays AI devices in rectangular containers, and data objects as cylinders [19].

2.1.2 Bonseyes Marketplace and the SVP

A key element of Bonseyes system is the Bonseyes Marketplace (MP). The stakeholders use the MP to prepare for their engagement in joint engineering. They use the MP to exchange AI artifacts and generate appropriate user licenses for them. The marketplace enables stakeholders to meet, offer and find artifacts, agree on terms and conditions for collaborations and use licenses, coordinate the exchange of artifacts, and, eventually, coordinate the use of an SVP. On the other hand, the SVP is a secure and protected PaaS where the computing tasks of AI engineering take place. The SVP provides the computing resources, storage resources, and AI artifacts accessible only by eligible stakeholders. It also provides means to shield the AI engineering from threats that can either be external, e.g., interception of information, and internal, e.g., misuse of artifacts. The Secure virtual premise (SVP) assumes market place (MP)

as a starting point for stakeholders to meet, find an artifact, and agree on terms and conditions for collaborative AI engineering. Meta information is exchange here i.e on which location to download the AI artifact or what is the general license for the AI artifact. License regulates the use of an AI artifact. Therefore, SVP enforces the use of licenses while using artifacts [19].

Figure 5 below gives an overview of the Bonseyes AI marketplace, Secure virtual premise, and User. The marketplace website consists of Frontend and the Backend components, the Frontend of the market place has two microservice React and Doc, while the Backend consists of five microservices: API gateway, Authentication, Community, AI Artifacts, and AI Assets. The frontend provides an interface to the end-users while backed provides data and functionalities through the API.

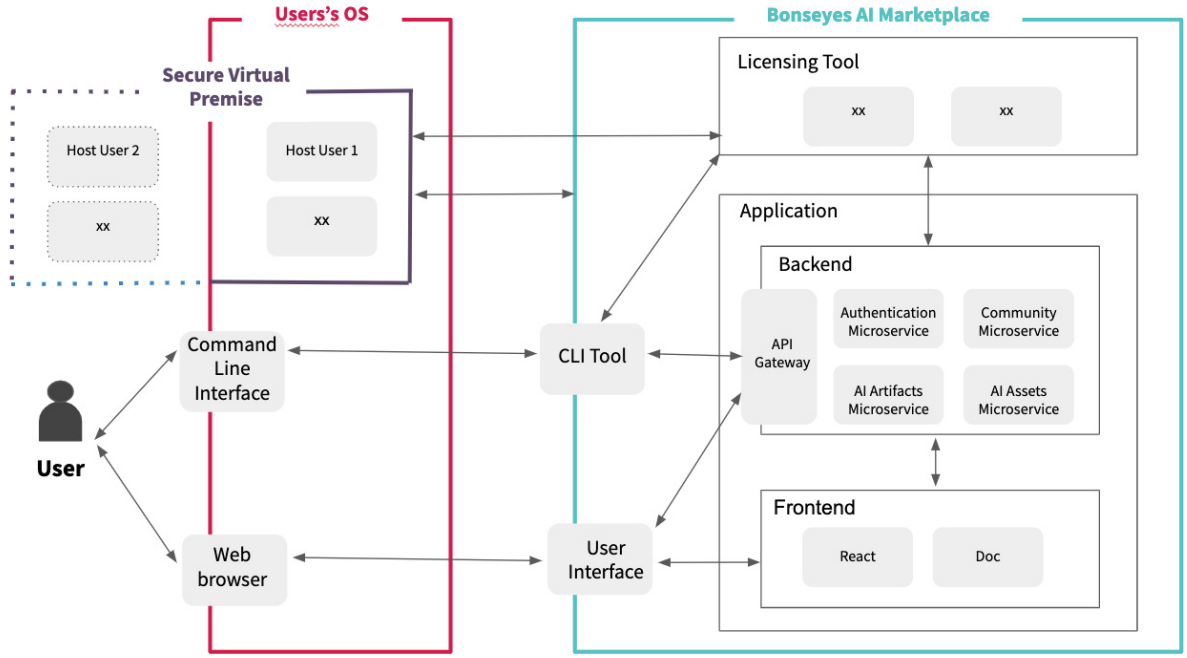

Figure 5 - Overview of the Marketplace architecture [20]

Users can access Bonseyes AI Marketplace with two client tools: a web browser and a Bonseyes CLI app. An online application can be accessed through the browser, which sources are executed in the browser that communicates directly with the AI Marketplace API, or as in an architectural distinction called Backend. The Bonseyes CLI Tool is run within the operating system console and interacts directly with the AI Marketplace API. The user may also directly use the AI Marketplace API to incorporate their automated workflows into the marketplace [20].

2.1.3 The Collaboration Model of SVP

The main collaboration model that is considered for the implementation of SVP is depicted in Figure 6. It is denoted as a coordinated consortium [19] because a group of people joins together in the engineering of an AI application and there is a distinct entity that is coordinating the coordinated consortium. This is the consortium of equal partners. The collaboration model of SVP describes the relationship of users in Bonseyes AI engineering architecture. From Figure 6, the AI artifact owners share their artifacts on the request of AI artifact users inside a secure virtual premise for AI engineering. The owner of the artifact is the initiator of collaboration, the owner and the user of the artifact collaborate inside the SVP. The current centralized SVP has limitations like scalability, reliability, and load balancing but on the other hand, it has advantages like consistency, efficiency, and affordability. The future version for the SVP is considered to be distributed and federated which will provide load balancing, high performance, scalability, and flexibility. The Coordinated consortium assumes roles and entities which are outlined next.

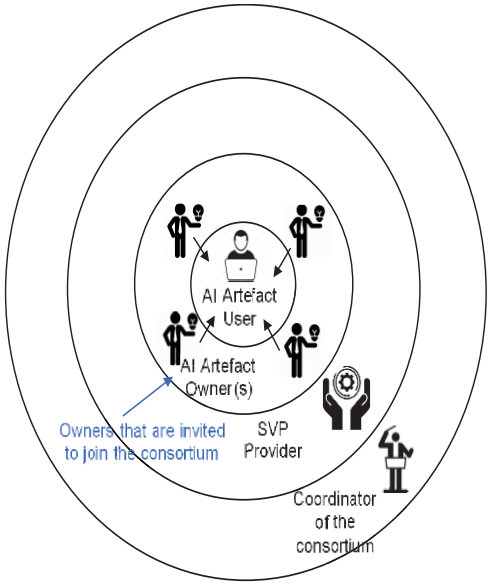

Figure 6 - Collaboration Model [19]

Coordinator

The coordinated consortium is organized and structured by a single entity called the coordinator. The coordinator invites different parties to take part in the collaboration process and builds trust to join the SVP. The coordinator is assumed to be trusted by all the stakeholders participating in the collaboration process.

SVP Provider

SVP provider organizes and provides the resources for implementing and executing the AI pipeline in the SVP. It is trusted by all the parties participating in the AI engineering process. The SVP provider enforces secure communication, execution, and coordination of users when using an artifact in a pipeline.

Artifact Owner

The artifact owner is the creator of the AI artifact and has the intellectual right on it. Artifact owner can be thought to be as Data scientist or the Data provider. The coordinator and artifact owner agree on the terms and conditions in the collaboration model of SVP to obey certain rules for using artifacts. These terms are mentioned in the end-user licenses.

Artifact User

The artifact user is the user of AI artifacts inside the SVP. Artifact user is AI Application Developer who buys AI artifact from the artifact owner in Bonseyes marketplace and utilizes it.

2.2 Logical Architecture of SVP

SVP is a secure virtual premise that enables trusted and secure collaboration among the stakeholders. The architecture of SVP is derived from its major aim i.e trust and security. SVP is based on the “edge computing view” which enables its implementation at ordinary company campuses and hospitals and avoids the use of centralized cloud infrastructure, such as large scale data centers. The logical architecture of the SVP [19] is depicted in figure 7. It consists of a single controller host, a computes host, and an AI artifact user. Owners and users of AI

artifacts meet at the Bonseyes marketplace. AI artifact Users select the artifact for AI engineering and agree on end-user licenses for the selected artifact with the author. Artifact owners and their artifacts are then invited to join SVP. SVP consists of multiple physical and logical entities and data objects:

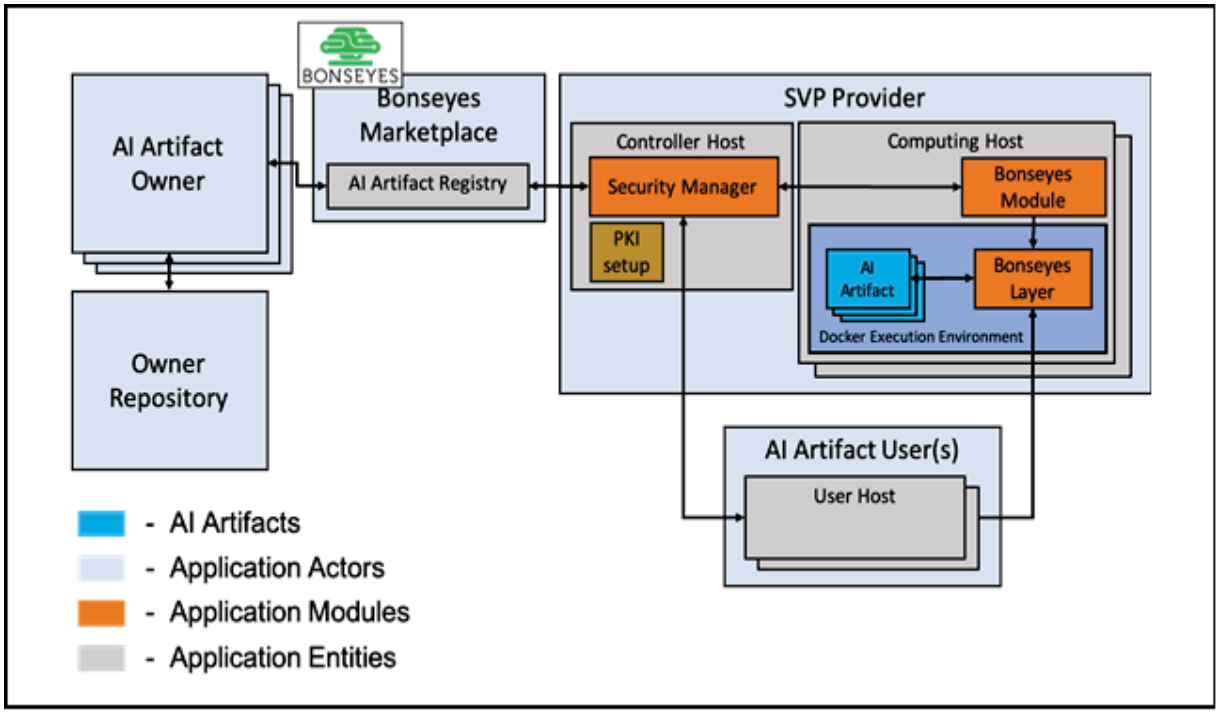

Figure 7 - Logical Architecture of SVP [19]

Controller Host

The controller host is a physical entity that executes a security manager. The current SVP implementation consists of only a single and dedicated controller host. It is managed by the SVP provider and is located at the SVP provider premise.

Security Manager

The security manager is a logical element located on the controller host. It acts as a technical trust anchor of the SVP. Security manager and related certificate authority build up secure communication with multiple users and compute hosts.

Compute Host

The compute host is a physical entity with the docker containers to execute AI tools. It also stores data objects embedded in an AI pipeline.

Bonseyes Module

Bonseyes module is a logical entity responsible for the authentication of the compute host against the security manager as well as initiates the bonseyes layer for shielding the AI artifacts. It makes use of a bootstrapper to start the Bonseyes layer container.

Bonseyes Layer

The Bonseyes layer is a proxy layer on top of the AI artifact. Bonseyes layer shield the AI artifact from misuse. It manages the command access to the AI artifacts. The commands are validated against the specification of the end-user license. The Command-line tool is used for the communication between Bonseyes layer and the AI artifact. It is initiated by Bonseyes module.

AI Artifact

AI artifact is, in general, a data or tool object which is present on the compute host. AI tools are enclosed in a docker container whereas data objects are stored on disk volumes controlled by the trusted compute host only. These volumes can also be set up on containers.

User Host

The user host is a physical entity and computer which is used by artifact user to access the artifacts inside the AI pipeline. The user host can access artifacts through Bonseyes layer using a command-line interface. The Bonseyes layer forwards the commands of those users who have valid license files for the AI artifacts.

3 RELATED WORK

3.1 Research Method

In this thesis, we used multiple research methods for providing solutions to the research questions. A literature review was conducted using keywords i.e Cloud federation and Federation of distributed resources to gain knowledge and understand the research problem stated in this thesis. A Graphical User Interface is designed and out of this design, we identified the requirements for implementing multi-location SVP. In Implementation, we applied an agile approach and it is carried out in a virtual lab which is Microsoft Azure. Our solution is structured in two parts i.e functional validation and numerical validation. In functional validation, we derive a use case and then we check whether all the steps in use case can be done. In numerical validation, we initiated one, two, and three locations within multi-location SVP and investigated the statistical analysis of start up time. In the end, we documented the answers to research questions.

3.1.1 Literature Review

The literature review aims to find a state of the art literature to gain knowledge and understand the research problem stated in this thesis.

Data sources and search strategy

The search process helps us to collect research papers related to our research topic. The search process involved the topics including Cloud federation and Federation of distributed resources. The standard online databases that we use for literature search are IEEE Xplore, ACM, Science Direct, Springer Link, Diva, etc. The result of the searches provided different papers which were analyzed by reading the title, abstract, conclusion, and connection of the papers to our research topic.

Inclusion and Exclusion Criteria

Inclusion and exclusion criteria were added to further narrow down our search to the research papers that are more relevant to our thesis topic. Inclusion criteria include papers that are written

in English, papers that are based on cloud federation, papers that are based on a federation of distributed resources, and papers that are based on cloud computing federation techniques. Exclusion criteria include papers that are not written in English and papers that do not address the research problem. Figure 8 depicts the different steps that were involved in our selection process of research papers.

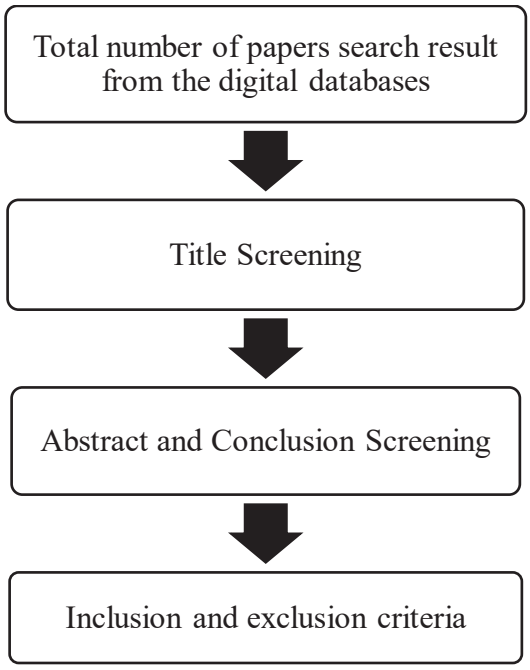

Figure 8 - Steps in Research Papers Selection

Data Extraction

Finally, data were extracted from the research papers that were selected. It provided theoretical knowledge about the research problem we are dealing with.

3.1.2 Design and Requirements Identification

A Graphical User Interface (GUI) is designed and from this Graphical User Interface, we identified the requirements for implementing multi-location SVP. A GUI design is based on the analysis of user workflow. Designing a Graphical Interface and deriving requirements from it is described in **chapter 4 – Method and Key Technologies, Section 4.2.**

3.1.3 Implementation

We implemented a framework derived from the requirements identified for multi-location SVP. In this experiment, we used the Microsoft Azure platform where the implementation was carried out and performance was tested. Here we applied an agile approach that relies on the spiral model.

3.1.4 Validation and Verification

Our solution is structured in two parts: a) Functional validation and b) Numerical Validation. In functional validation, we derive a use case from the AI workflow and collaboration model of SVP and then we verified whether all the steps in the use case can be done by what we have implemented. In Numerical validation, we verified the performance under a simplified scalability test. For statistical analysis, we initiated one, two, and three locations within one multi-location SVP and measured the average start-up time. Functional and Numerical validation are described in **Chapter 5, Section 5.5, and 5.6.**

3.2 Related Work

This section presents the related work that was done that becomes the underlying support for our thesis. We present the researches that use federation techniques for combing distributed resources.

In [7], the author presents the concept of a global environment for network innovation (GENI). GENI is a distributed virtual laboratory for carrying out experiments in network science, services, and security. It is a virtual testbed concept and is suited not only for testing but also for the deployment of new network architectures. GENI provides a federated venue that permits a growing number of experiments in a realistic deployment environment. GENI uses technologies like SDN and GENI racks in support of GENI campuses and application to achieve shared, deep programmable and national-scale distributed architecture. The key features of GENI [8] are sliceability and GENI aggregate. Sliceability is the ability to support virtualization and a chain of resources. A GENI slice is an isolated environment where the experiment takes

place. A container contains resources that are used in the experiment. These resources (compute, network links, etc) are added by the experimenters. Resources that are added to the slice take part in the experiment. Members who are added by the experimenter can only make changes in the experiment. The experimenter who creates the slice decides who can or cannot join the slice. The resources that are provided to the GENI experimenters are by GENI aggregate. The GENI aggregate is a place where multiple resources are located. An experimenter requests an aggregate for resources. GENI aggregate is storage for resources. These resources are arranged in the GENI racks present at a different location. There are different kinds of resources in different aggregates. Some contain compute resources, virtual machines other may provide network resources to connect distributed computing resource present at different aggregates for the experiment. An experimenter requests the resources from aggregate using aggregate manager API.

The paper in [9] presents a decentralized and distributed system that units enterprise clouds, networks, and peer-to-peer techniques for better computing. The Aneka federation combines different Aneka enterprise cloud services and nodes distributed over multiple domains acting as a single site. The Aneka coordinator is a resource management and resource discovery tool employed in an Aneka Enterprise Cloud to speak and share resources with different Aneka sites. The Aneka coordinator consists of the Aneka Services and Aneka Peer parts which give the Cloud's core ability to act with different services. Aneka Services provides useful peer-to-peer programming and peer-to-peer execution whereas the Aneka Peer facilitates resource sharing and load reconciliation among the distributed Aneka Enterprise Clouds so providing a redistributed Infrastructure as a Service (IaaS) federation.

In [10] the author presents Empower, which is a collaborative research and experimental testbed for software-defined networking (SDN) and network function virtualization (NFV). The paper discusses the challenges in implementing a fully virtualized experimental facilities supporting NFV in the wireless networking domain and Wi-Fi-based networks with the Empower experimental testbed. Furthermore, the author discussed the requirements for the Empower design and presents the Empower architecture. Builds on top of the SDN system for Wi-Fi networks integrating OpenFlow with control of open energy usage and Toolkit Management.

4 METHOD AND KEY TECHNOLOGIES

4.1 Agile-like and Incremental Design

This section describes our approach towards the architectural design of distributed secure virtual premises (SVP). The SVP aims to protect and securely integrate the collaborator's AI artifacts and tools into shared AI pipelines for AI engineering. This thesis focuses on distributed SVP, which must satisfy the requirements of multiple locations, more users, and multiple artifacts for trusted and secure AI development. Therefore, we used a design methodology that relies on the spiral model depicted in Figure 9 inspired by [11], which facilitates iterative development. The spiral model has shown success previously in infrastructural design projects like GENI [8].

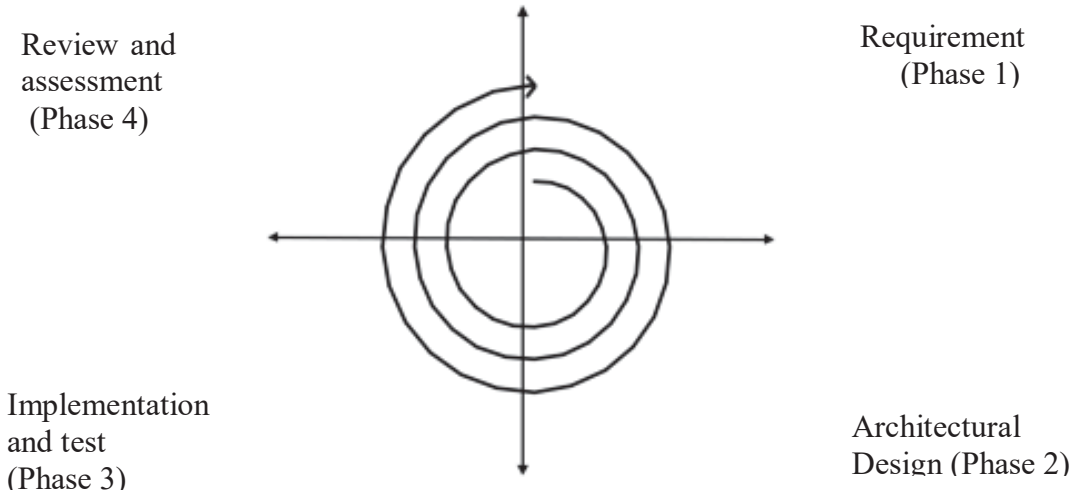

Figure 9 - The Spiral model of Architectural Design [11]

In the spiral model, each revolution compromises of four phases of system design development. The first phase of revolution identifies the system requirements. The second phase consists of architectural design concerning identified requirements. The third phase represents the implementation of an architecture that needs to be tested which is subsequently evaluated in the fourth phase. However, new requirements might arise during phases three and four which need to be addressed in the next iteration.

4.2 GUI Design by Mockup

The design of the GUI is based on the analysis of the user workflow. Hence, one has to identify the user's workflow abstractly, identify the major steps, and derive a mockup for the GUI for these steps. In general, the design uses three steps: 1) it starts with the identification of the high-level workflow, 2) abstracts these steps and decide which step is addressed by the considered part of the GUI, and 3) design a mock-up of the GUI that meets the major needs of these steps. If we look at pipeline 1 in figure 4**, Chapter 2, Section 2.1.1** let's say the task of the user is to configure this AI pipeline and the AI tools can be at various locations. The high-level workflow is to define the pipeline and then to define the locations of the elements of the pipeline and then to configure distributed SVP. The second step is to decide how these steps can be addressed by GUI and the last step is to design a GUI for these steps.

Bonsey's uses machine learning AI pipelines to automate machine learning workflows. The AI pipeline consists of several modular steps (tools) to train a model or to generate an application. The user and AI artifact owner meet at the Bonseyes marketplace. The user selects the AI artifact to be reused in the AI pipeline and agrees on the end user license for the selected artifact with the owner of the artifact. The user would now need to configure a distributed SVP where he can place these AI artifacts for the execution of the machine learning AI pipeline.

.

To execute or orchestrate the AI pipeline inside SVP, the user requires a webpage or GUI from where he can select and start multiple SVP locations to perform AI engineering. Figure 10 shows the GUI designed by mockup for distributed SVP.

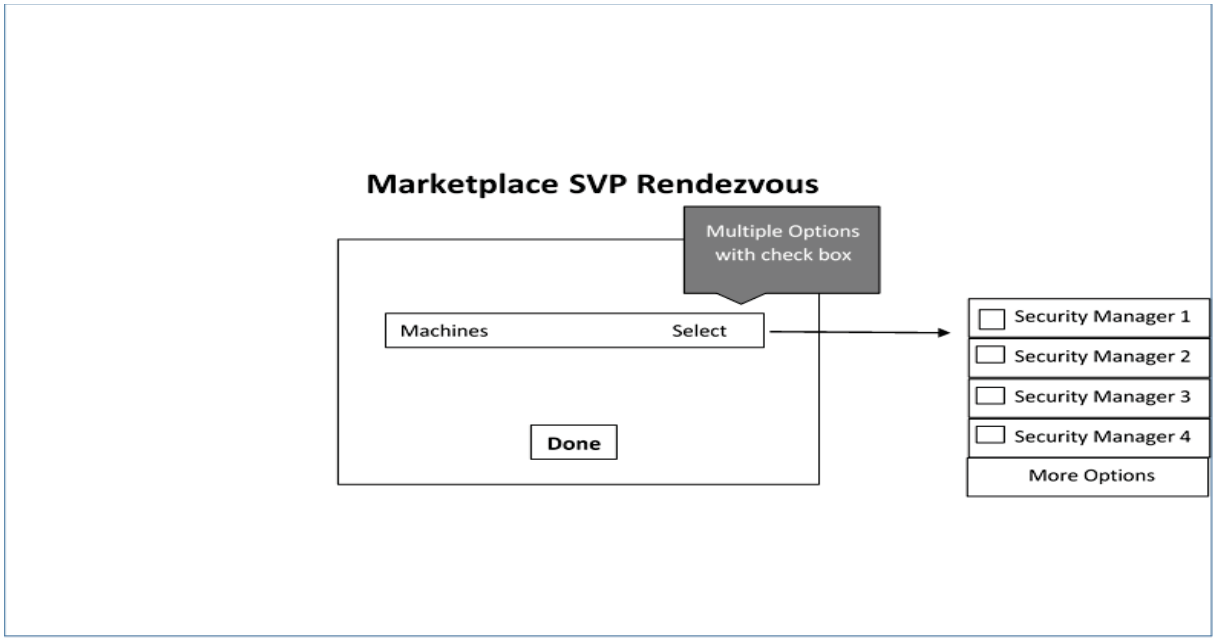

Figure 10 - Distributed SVP GUI Mockup

Marketplace SVP rendezvous host provides an interface for the users to select execution infrastructure for AI engineering. This interface consists of multiple selections drop-down list box with multiple security managers. This interface is considered as an access point for the users to execute resources. The users can select the required number of machines and after the selection, it can initiate the selected machines for distributed AI engineering.

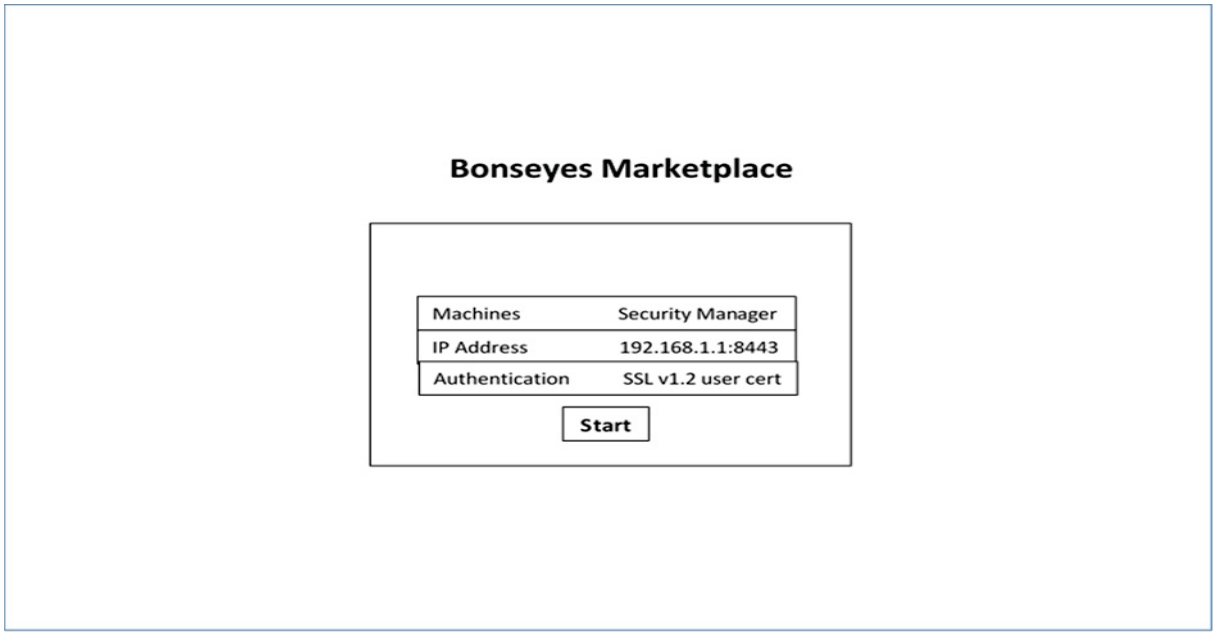

Figure 11 - Centralized SVP mockup

Figure 11 represents a single SVP location mockup design which is already developed and deployed by the Bonseyes project. There is only one single security manager, its IP address, and a user certificate to authenticate against the Bonseyes module.

4.3 The Microsoft Azure: Experimental Platform

This thesis requires a lab where the mechanisms for the multi-location SVP are implemented and tested to provide lab-based computational power, computers, and networking. To use this in a scalable way the Microsoft Azure platform was chosen as an experimental platform where the implementation was carried out and performance was tested. Next, we briefly explain what is Microsoft Azure and how it is used in this thesis.

In traditional data centers, the users have to handle and manage everything like purchasing and installing hardware, virtualization, operating systems, required applications, setting up the network, configuring the firewall, and storing data in the storage. Once all this is done, the user has to look after everything throughout its life cycle. This forces a great cost for the hardware and as well as operational cost for its maintenance. It gives the users the freedom to select any hardware and software of their choice but they also have to pay for it whether they are using it or not.

Cloud computing is the replacement for the on-premise datacenters. A public cloud vendor is responsible for the buying and maintenance of the hardware. Usually, they provide a large number of platform services that a user can use. User can use any hardware and software required, that might be very expensive, by leasing them on demand. This results in saving the large amount of money that is spent on the locally built datacenters hardware and just pay for the operational expense of the underused services. The cloud vendor only charges the users when their hardware or software is in use and not in the idle mode.

Microsoft Azure [12] is a public cloud developed by Microsoft for building, testing, deploying, managing, and offering services and applications. Microsoft Azure is a platform that provides software, platform, and infrastructures as a service. It comes with an online portal from where a user can manage the compute, storage, network, and application resources. A user can select virtual machines using the portal and specify the computing power like CPU, RAM, and local

storage. Furthermore, a user can also specify the desired operating system, network configuration, location of the node, and software. After all the specifications the user then can deploy the VM and can access it within the minutes.

Microsoft Azure provides a large number of services of which we are mentioning a few of them here. The compute service of Azure includes Azure virtual machines, Azure Websites, and mobile services. The data service of Azure includes file services, SQL database, Queue, Blob, Table, etc. Application service of Azure includes azure active directory, azure media service, azure scheduler, etc. Network services of Azure include Azure virtual networks, content delivery networks, and azure traffic manager.

In this thesis, we used Microsoft Azure Infrastructure as a service in the experiment part. This enables us to create requirement specific virtual machines running on the Microsoft Azure infrastructure. On each virtual machine, we installed Linux 18.04-LTS operating system and specified 2 vCPUs, 16 GB RAM, and 30 GB of storage. This service does not offer control over the hardware or virtualization software but gives you control over other things. Microsoft Azure is responsible for the handling and maintenance of the hardware and virtualization software residing in the Microsoft datacenters. There is another big advantage of using the Microsoft Azure IaaS platform is that you can easily migrate the files to and from Azure VMs to the VMs that are present on local computers or data centers. Figure 12 below depicts the Microsoft azure portal representing different services.

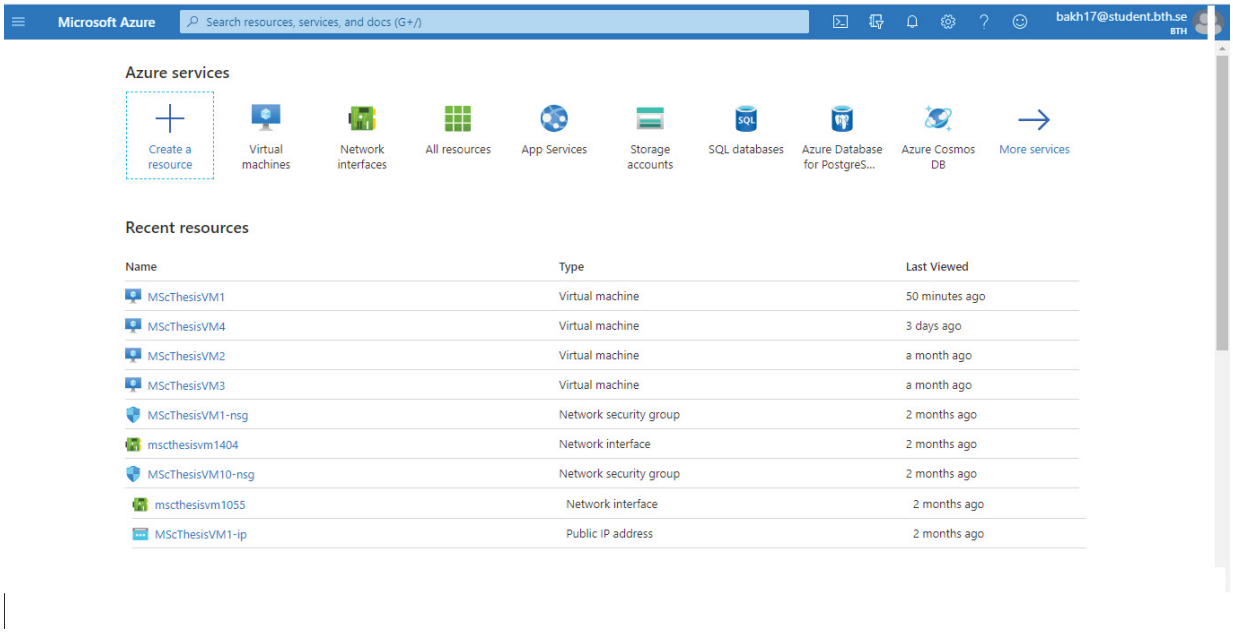

Figure 12 - Microsoft Azure portal [12]

We used multiple Microsoft Azure VMS in our experiment. To access the Microsoft azure VMs, we need to login to the Azure portal with a username and password. We used the SSH protocol to connect to remote VMs and installed required applications on them.

4.4 Certificate Management

Certificates govern the access, authorization, and authentication of the users when using the artifacts. Also, communication between the different elements is HTTPs based and therefore certificate management is an important part of this work. Next, we briefly explained the Certificate management, PKI, and digital certificates.

Certificate management is the process of monitoring, facilitating, buying a digital certificate from the certificate authority and signing it, installing it on the endpoint, renewing it after it expires, or revoking the old one and requesting a new one from CA for uninterrupted network operations. Every system that is connected to the internet needs a certificate for secure communication with another system. Companies or organizations that are assigned to be an administrator for managing PKI has to deal with a large number of certificates. These certificates need constant monitoring to be effective. A proper certificate management strategy

can help the administrators in the long run to prevent downtime and outages of endpoints caused due to expired or faulty certificates.

Public key infrastructure PKI [13] is a framework that is used for the public key encryption and to establish secure communication between the server (website) and client (user). PKI uses public and private keys for encryption and decryption of data when sending across the internet. A user who connects with the website gets the public key while the private key is generated when a connection is established and remains secret. Users use the public key to encrypt and decrypt the data when communicating with the website and the server uses a private key. This is how data is protected from changing and stealing. These keys are not only important for the encryption or decryption of the data but also helpful in the authentication process to verify the user to whom encrypted data is sent or to identify the communicating entities or devices.
PKI certificates [14] are the digital identities of the endpoints that are taking part in secure PKI communication. Certificates are like digital identities for websites. These PKI certificates are known as digital certificates. A digital certificate contains the certificate holder name, serial number, expiration date, a public key of the entity, and certificate issuing authority signature. The public key is shared with the user through a digital certificate and the private key is only known to the owner of the certificate. When a file is encrypted with the private key it can only be possible to decrypt with the public key. The file which is decrypted with the public key assures that the intended parties are communicating. A digital certificate is certified by the trusted source, which makes sure the entity is who they claim to be. That trusted source is a certificate authority. A certificate authority is an organization or company that issues certificates and digitally sign them using a private key which assures the certificate authenticity. Certificates that are not issued by the legitimate CA's are not trusted by the web browsers and they stop the access to such websites that are protected by such certificates. The registration authority is the one that verifies the identities of the clients who requests the digital certificate from the certificate authority. Furthermore, there are three types of certificates: TLS/SSL certificate, Code signing certificate, and client certificate. TLS/SSL certificates also known as server-side certificates are installed on the server. These certificates ensure secure communication between client and server by making it private and encrypted. The server can be a web server, app server, or any server that requires authentication for sending or receiving encrypted data. When a client tries to connect to a server, it first checks the server certificate. The client generates a session key which then encrypts with the server public key. The server

uses its private key to decrypt this session key. This is how a secure session is established between client and server. Code signing certificates are files used to sign software or files to verify the identity of the developer and ensures the credibility of the files. It assists the users to decide whether the software or files can be trusted. Client certificates are used to recognize a user or machine and it authenticates the sender and receiver.

In this thesis, we used PKI technology for the authentication and authorization of the users when using AI artifacts. The communication is conducted by HTTPS and web servers are installed inside each security manager, Bonseyes modules, and Bonseyes layers. A PKI.sh file is executed on each security manager. It performs the following tasks: a) Generating the key and certificate for Root and intermediate CA, b) Generating the self-signed certificate for Bonseyes Root, c) Generating the Certificate Signed Request(CSR) for ca1, d) Signing the ca1 certificate by root, e) ca1 is used to create the certificate for all entities, therefore, we create the chain of trust for ca1, f) Generating the Security Manager certificate, g) Generating the Bonseyes Module certificate h) Generating user certificates for User, Security Manager, Bonseyes Module and i) Generating .p12 certificate for User. The secured communication is enabled by the PKI infrastructure of the SVP.

4.5 Virtualization Environment: Docker

The main virtualization environment which is considered here is docker. Next, we briefly explained the Docker and the requirement of the technology in this thesis work.

In computing, virtualization is the technology that lets you create the virtual of something rather than actual hardware such as virtual servers, networks, computers, applications, and storage. It simulates the functionality of physical computers and creates a virtual computer system [15]. Virtualization optimizes and increases the operational flexibility of hardware resources in cloud infrastructure. It provides the separation of the different parts of applications and services by creating a virtual layer between them. There are three types of virtualization techniques in cloud computing. These are full virtualization, paravirtualization, and container-based virtualization [15]. Full virtualization and paravirtualization techniques use a software called hypervisor which interacts with the host physical instance CPUs and storage disk enabling the isolation of

the guest operating system instances. Running multiple guest instances with their own operating systems demand a large number of resources. That led to high CPUs, and disk space requirements, and low performance. On the other hand, container-based virtualization does not use hypervisor software. This type of virtualization is also called containerization. Containers are the product of the feature of an operating system kernel and they are isolated from each other. They run as a process on the host operating system which makes them very lightweight and requires fewer resources while running multiple instances [16]. Each container gets limited and controlled resources. Every container has a unique process id and file system namespace.

Docker [17] is a lightweight and open-source container-based virtualization technology. It was created using go programming language. It works on top of the underlying operating system and infrastructure. Docker uses kernel namespaces and cgroups of Linux kernel to limit resources and isolate its processes from the host operating system.

Docker makes it easy for programmers to create, deploy, and run applications using containers. It permits the developer to pack the application with all its dependencies and libraries and deploy it as one package. This makes the application run on any Linux system regardless of the machine settings that could differ from the machine used for writing and testing the code [18]. Figure 13 depicts the architecture of docker virtual technology. It shows the multiple lightweight containerized applications sharing a single host operating system.

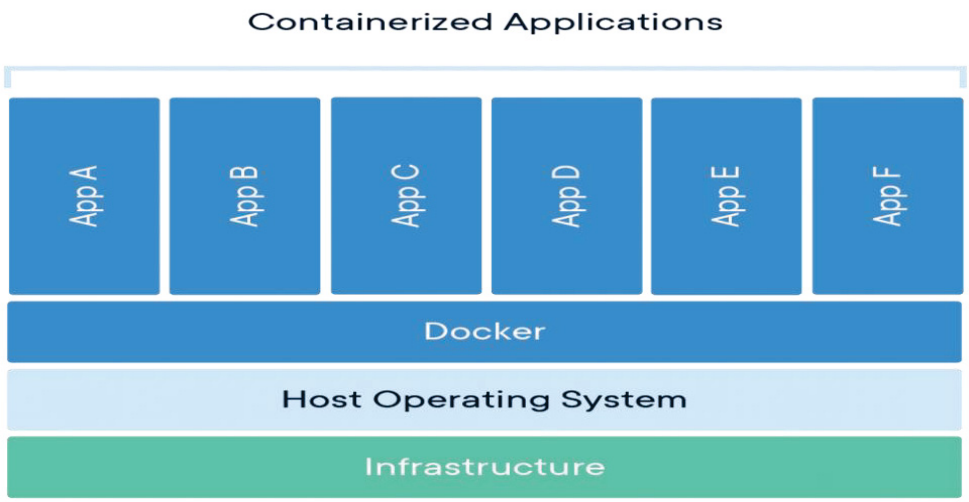

Figure 13 - Architecture of docker virtual technology [17]

Docker follows the client-server architecture. It consists of three components: docker client, docker server, and docker registry. Docker client interacts with the docker server through REST API. Docker server or host consists of a docker daemon which handles the request from the client. When a client request service using docker build or docker pull command, the daemon serves that requests, and it also manages objects like images, containers, volumes, and networks. Docker registry is a place where docker images are being stored. Docker hub is a public registry, where anyone can look for images by default and can use it. Docker image is an object that consists of a read-only template with instructions to build containers. Dockerfile can be used to create a docker image.

The Bonseyes project is using the docker containers to shift from large scale Cloud servers to edge computing. This fulfills the demand of computing resources at the edge and makes it flexible and elastic. Containers have an advantage over VMs due to their lightweight and portable runtime. They can easily develop, test, and deploy applications to servers and can interconnect. They are easy to install and are fault-tolerant which makes the resources at the edge reliable. In Bonseyes, the secure virtual premise consists of the Bonseyes layer to protect and interact with the AI artifacts. Bonseyes layer exists in the docker environment of the compute host and is initiated by the Bonseyes module. Bonseyes layer is implemented as a docker container. It is the only logical object that can connect and forward commands to the AI artifacts. AI artifacts are also encapsulated in the Docker containers.

5 SOLUTION AND ANALYSIS

5.1 Requirements from User for Multi-Side SVP

The secure virtual premise concept is based on the idea of defining a protected virtual area and hence the name "premise" where stakeholders can trustfully meet and collaborate. The land tract considered here, however, is not a physical area or a single dedicated physical compute infrastructure but is an emulated space consisting of interconnected and virtualized computing and storage resources. Also, the SVP can be started on-demand, i.e. with a high degree of automation and only when needed [20].

From a user perspective, SVP is a secure area where collaborative AI engineering takes place. For multi-side SVP, the requirement is to execute AI artifacts trustfully inside SVP at multiple locations and to guarantee protection against any unauthorized use while maintaining the distributed nature of the AI engineering. In Conventional Cloud computing, centralized computing locations are used, which can be protected by centralized components like firewalls. Whereas SVP uses a proxy layer that must be executed successfully at various locations. A layered concept is used to shield AI artifacts. This layer is denoted as Bonseyes layer. Bonseyes layer is used to protect, control, and filter access to the AI artifact. It acts like a proxy layer and all the incoming communication to AI artifact is received by BL. Bonseyes layer is the only entity entitled to talk to AI artifacts using Bonseyes CLI tool.

To execute or orchestrate the AI pipeline, the user needs a webpage from where he or she can select the resources to use and then execute. A tool or a webpage is needed, which should enable the user to select multiple SVP locations for locating AI artifacts. After the selection of multiple locations, the user should start, control, and stop these locations for distributed collaborative AI engineering. Figure 14 below represents the mockup of the webpage to enable interaction between users and processes. The design is based on a generic pipeline structure and a generic structure of the SVP which considers various locations where AI artifact can be executed. A Webpage represents a centralized entity from where these multiple remote computing resources can be initiated. Now that a user can successfully start the remote SVP locations, the elements of SVP require a secure mechanism to communicate with each other. This secure communication can be enabled by the PKI infrastructure of SVP. HTTPs protocol can be used for reliable

communication between the entities of the SVP. To enable secure and reliable communication, inside each Security manager, Bonseyes module, and Bonseyes layer of multi-location SVP, a web server needs to be installed. This will enable the use of server certificates on each of these entities. which will ensure reliable and encrypted transport of structured data among the SVP elements.

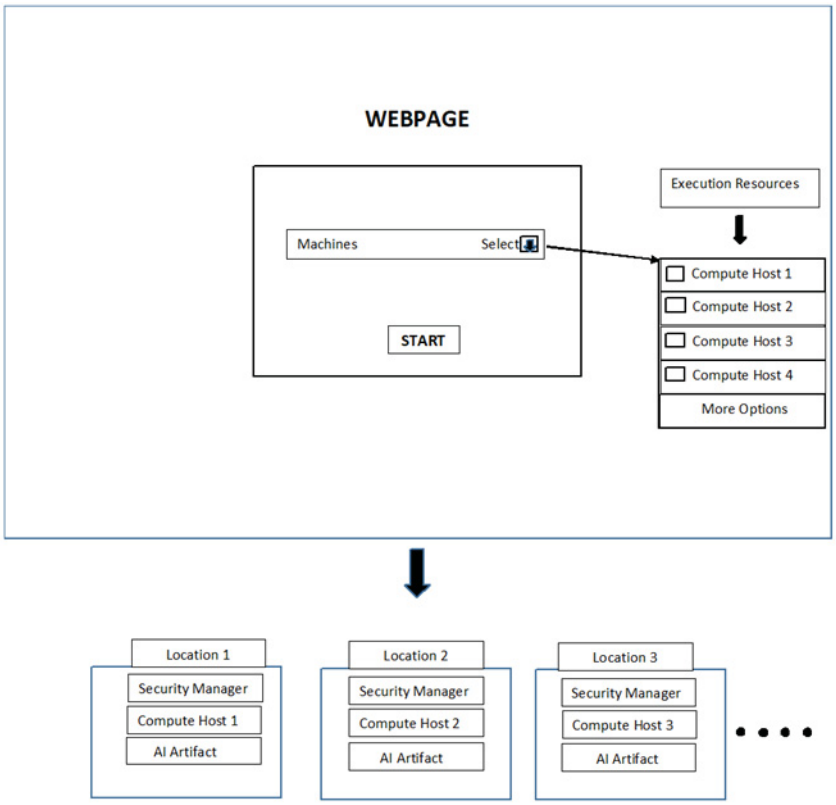

Figure 14 - Mockup of webpage and processes

5.2 Overall Architecture of Multi-Location SVP

An initial model solving the requirements and aims of the distributed SVP concept and its capabilities highlighted in Figure 1 is explained and depicted in Figure 15. The figure shows that an involved user can request different services/software artifacts for running an AI algorithm which is provided by the software developers represented by the locations. The SVP is the secure space where the collaborative and automated engineering, execution, and computing of AI pipelines are performed. Inside SVP a controller hosts and compute hosts are

present. Every compute host consists of Bonseyes layer whose role is to provide an interface towards AI workflow, check and validate the enforcement of the licenses and handle the resource infrastructure that is used for the storage and execution of software artifacts. SVP also consists of a Bonseyes module which further consists of Bonseyes layers and the software artifacts which are placed in a docker execution environment.

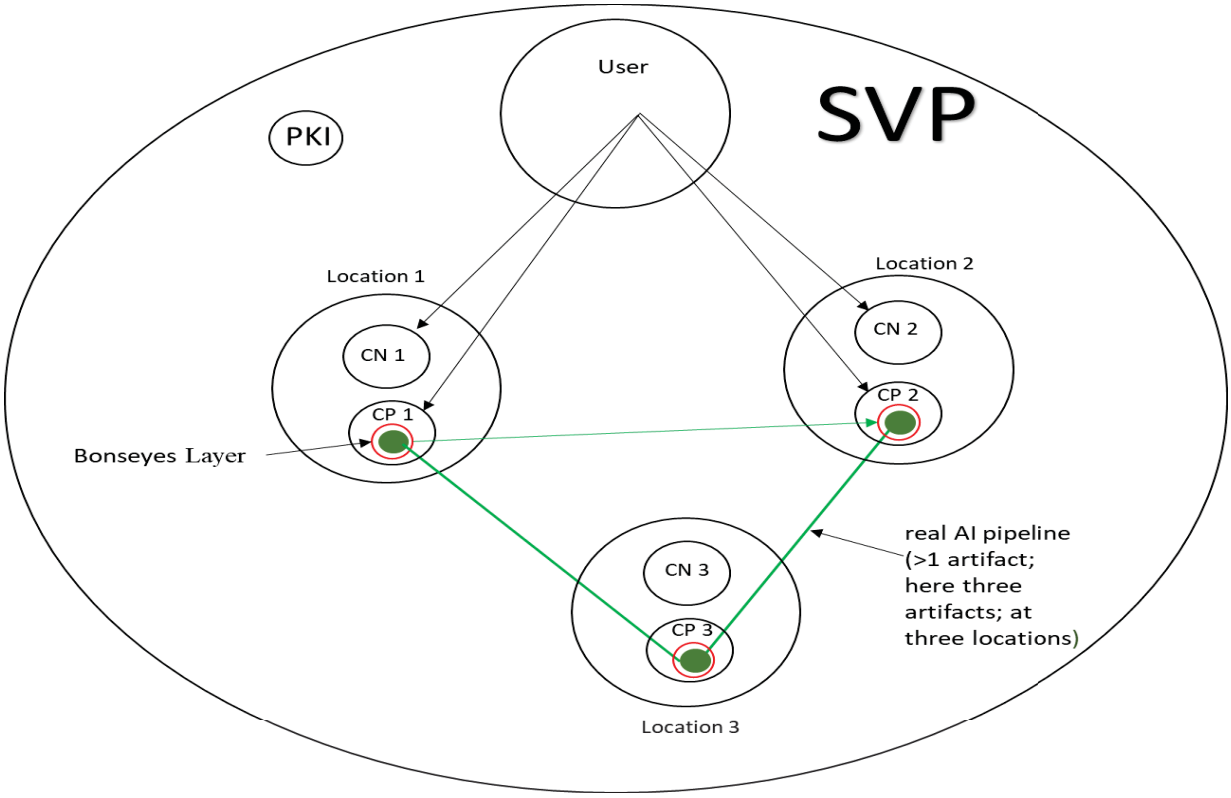

Figure 15 - Federation of different locations into one SVP

Distributed SVP infrastructure

Figure 16 shows the enhanced architecture of multi-side SVP. An SVP user can select from the multiple remote execution resources or compute host at an SVP Rendezvous Host for the secure execution of AI artifacts. Furthermore, the SVP Rendezvous Host manages user profiles, including their cryptographic key and certificates. Communication between the user's host and remote execution resources is based on the HTTPs protocol. Therefore, each infrastructure entity has its certificate and keys issued by SVPs PKI certificate authority. Users enable secure

communication by registering with Bonseyes marketplace. Marketplace checks the user information and after verification it requests the Bonseyes Certificate Authority to issue a certificate for the accepted user. The certificate is handed to the user and the user installs it in the web browser. Connection to the SVP's APIs for the implementation, orchestration, and control of AI assets at remote locations is regulated and checked for compliance with the guidelines outlined in an AI asset license file. Only the authenticated and approved users can use an AI asset's APIs [20].

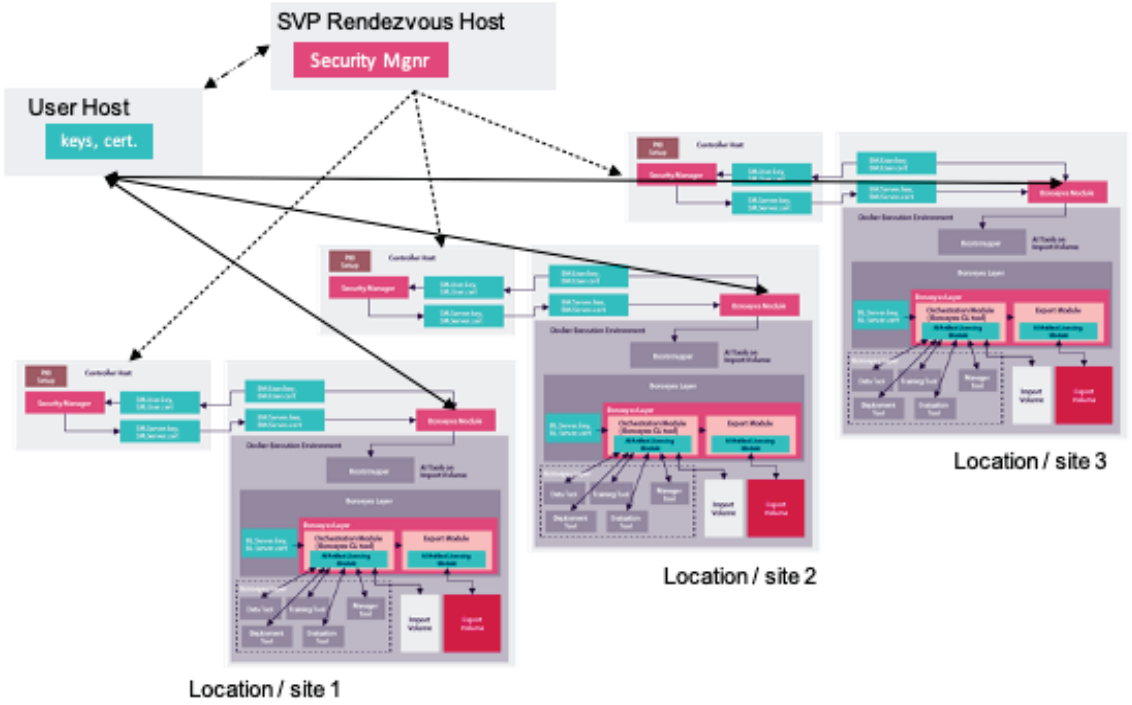

Figure 16 - Enhanced Version of the SVP for federated, multi Location/site SVP computing using an SVP Rendezvous Host [20]

The detailed architecture of SVP individual compute host, controller host, the exchange of cryptographic keys and certificates, and its relation to an AI artifact repository is depicted in figure 17. SVP uses HTTPs based secure communication among its elements and with the users. The SVP's central feature is the Bonseyes Layer, which is deployed as a Docker container and offers artifact shielding. The second essential element is the Bonseyes Module (BM) which resides on the compute host. It begins the trustworthy Bonseyes Layers modules for the AI objects which exist in the compute host's Docker environment. The collection of BLs for a specific pipeline forms the SVP. Eventually, the layers are applied through various compute hosts, allowing federation. The Bonseyes Module makes use of the Bootstrapper to launch the Bonseyes Layers containers. This achieves a generic initiation of arbitrary Bonseyes Layers.

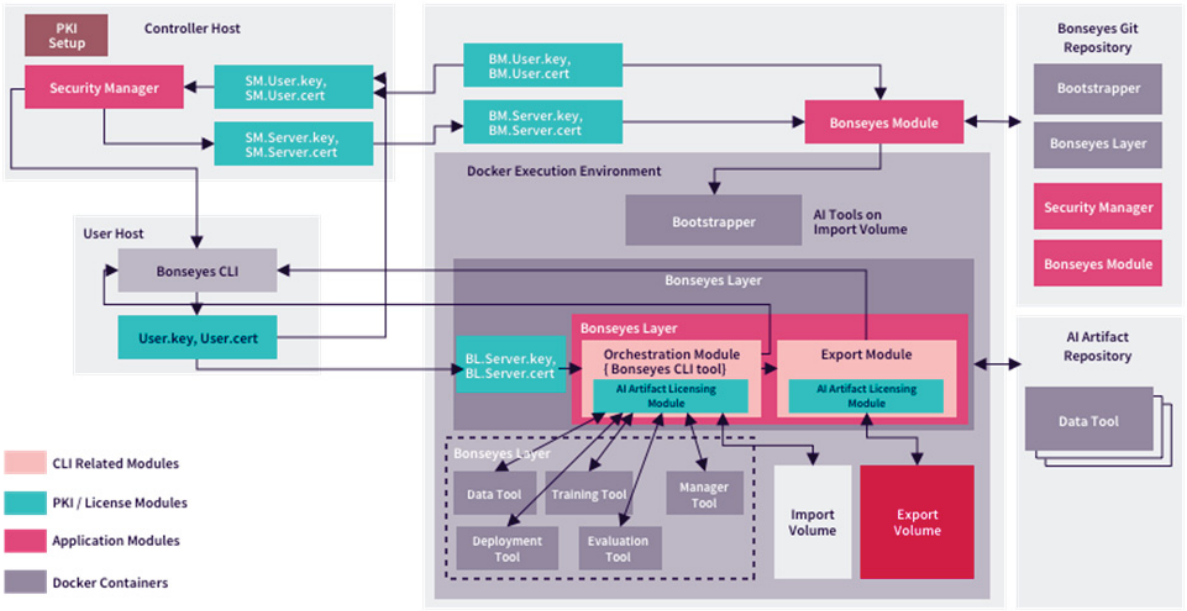

Figure 17 - The detailed architecture of the SVP on the compute host showing also the exchange of cryptographic key and certificates [20]

The Bonseyes layer shields the AI artifacts and is the only entity that can interact with the AI artifacts. All the commands are received by the Bonseyes layer and then forwarded to the AI artifacts. Bonseyes layer can filter the commands and verify these commands with the agreed user licenses. The orchestration module inside Bonseyes layer matches every command received for the AI artifact with the agreed user license file inside the AI licensing module. If the command received for the AI artifact matches with the provided license, then the orchestration module passes the command to the AI artifact. The AI Artefact Licensing Module compliance verification assumes that a valid license has been submitted to the SVP and the Bonseyes Layer to which the Bonseyes Layer serves as a proxy.

The Export Module provides the interface for uploading and extracting data from and into the SVP and to locate this information so that it can be interpreted by a Bonseyes Layer and related AI objects. The module describes tracked volumes of input and output, i.e. file systems on which data can be stored and recovered and which have monitored access. As the upload and download feature is identical to the AI artifact or an AI pipeline orchestration, the Export

Module carries out compliance verifications identical to those of the Orchestration Module, i.e. it verifies upload and downloads commands against a license [20].

5.3 Implementation and Use of Agile Design

The implementation of multi-location SVP is achieved in two iterations. In the first iteration, we deployed and tested the working of centralized SVP on Azure virtual machines. The centralized SVP infrastructure consists of a controller host and a compute host [19]. The controller host consists of a security manager which is responsible for the technical security of SVP. The compute host consist of Bonseyes module and Bonseyes layer which exists in the docker environment of the compute host. We deployed controller host and compute host on separate Microsoft Azure virtual machines. We reviewed and assessed the running SVP. After successfully running the SVP, in the next iteration, we addressed the enhanced SVP.

In phase 1 of the second iteration, requirements were derived from the graphical interface design by mockup as described in **Chapter 4, Section 4.2** to achieve multi-location SVP, enabling multiple users, artifacts, and locations for collaborative AI engineering. After understanding the requirements, in the next phase, we designed the architecture for the multi-location SVP as described in **Chapter 5, Section 5.2**. In phase 3, we implemented the framework of multi-location SVP and verified its functionality on azure virtual machines and tested it. We implemented the marketplace rendezvous host with the graphical user interface. The backend of the rendezvous host consists of a webserver and SQLite database. We build web APIs for requesting different services like starting multiple remote execution resources, stopping these resources, and adding new resources. We implemented Marketplace rendezvous host on localhost and multiple location SVP controllers and compute hosts on Azure virtual machines. In the last phase, we reviewed and assessed its working, to satisfy the requirement of this project.

5.4 Implementation of Start and Stop Sequence

One of the objectives of this thesis work is to start and stop multiple compute hosts for AI engineering on user demand.

Marketplace SVP rendezvous host is the entity with the GUI to Start, Add and Delete multiple distributed remote locations. It is a gateway for initiating multiple SVP compute hosts. Marketplace rendezvous host consists of web APIs for interacting with different microservices. The start API has a sequential script. The sequential script used here is a python loop, to start multiple SVP compute hosts. When a user selects multiple machines and presses the start button, for loop starts and requests the IP addresses of the security managers associated with compute hosts stored in the SQLite database and opens them in a new window tab one by one. It iterates over the same block of code again and again until all the selected machines are started. Each remote location starts up one after the other.

The implementation of starting multiple SVP locations is done sequentially. The start sequence is input/output bound which means it only depends on the speed of the input/output subsystem. If we look at the multi-side SVP infrastructure, each server we are using is running on a dedicated machine. The SVP rendezvous host is a separate entity hosting web API's to interact with different services. These API's are very lightweight and does not require too much processing while requesting multiple services at a time. However, it is also possible to start multiple locations in parallel or concurrently. In practice, when a loop runs, it iterates over the block of code to execute each step one by one which means there is always only one process executed at a time. The execution of one step at a time is considered slow, less CPU utilization, and assures execution in each repetition. On the other hand, the parallelization allows concurrent or simultaneous execution of multiple processes. Parallel execution of a process is considered fast because it cut through the workflow, maximum utilization of CPU, and execute the processes altogether but it has some drawbacks like it requires synchronization between different processes and another problem is asynchronous error management. The parallel execution of a process also requires multiple cores which results in excessive power consumption. This system is designed to be used on edge devices like Raspberry Pi which requires very little computation power to execute AI tools. Both sequential and parallel processing have advantages and disadvantages

The stop sequence is implemented on the individual Security Managers of distributed SVP. when an artifact is started it starts with a random name. when we want to stop it, the docker requires the name of the container because it is random. We get the status of all running containers using docker ps -a command, from this output we find the name of the container

corresponding to the artifact image name. Then we use the docker stop (container name) command to stop the artifact container.

5.5 Functional Validation

Use Case

To use the multi-location SVP, as a user, we need to agree on the end-user licenses for the artifacts at the Bonseyes marketplace with the owner of the artifacts. The artifacts owner and the artifacts are then invited to join SVP and these artifacts are then embedded in the corresponding pipeline. Let's say now we would like to configure assuming a very simple AI pipeline 1 as shown in figure 4, **Chapter 2**, **Section 2.1.1** which consists of three AI tools, and all of these AI tools will be at different locations. Now we need to select three hosts and on each of these locations, we would like to execute these tools.

Validation by Testing whether a Workflow can be Executed

The user would first require to register on the SVP rendezvous host. Then after creating the account, the user would enter the username and password in the specified fields on the webpage to sign in as shown in figure 18.

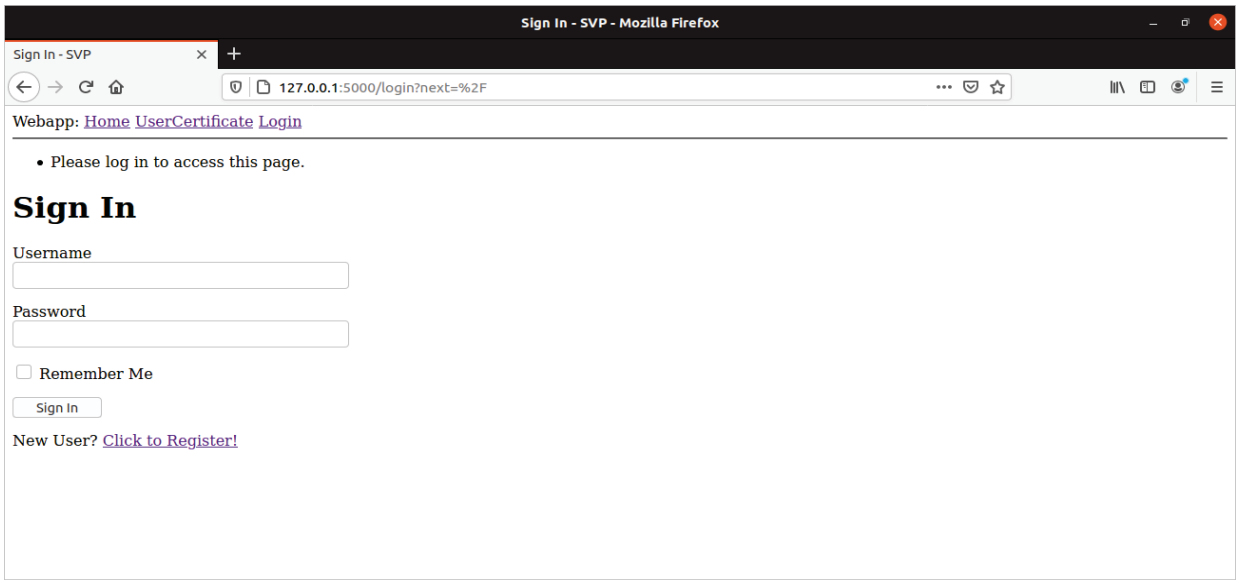

Figure 18 - User Login Page

The data related to the user is stored in the SQLite database. SQLite database consists of the user table. The data in the user table is stored as rows that consist of fields like id, username, email address, and password hash as depicted in figure 19.

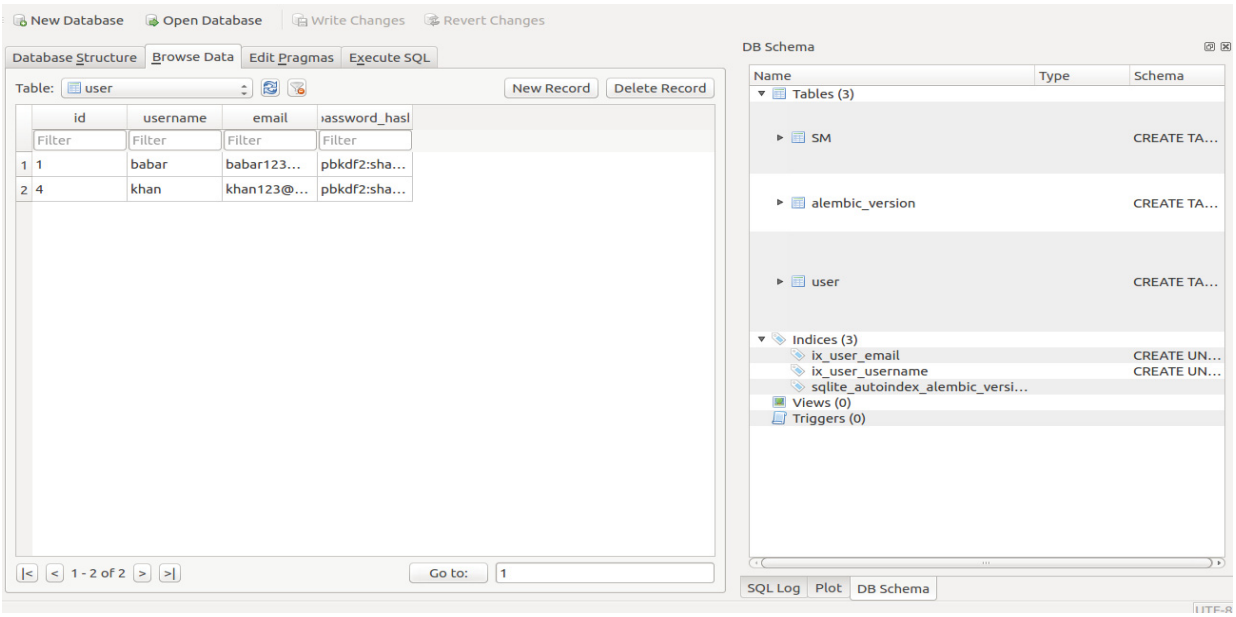

Figure 19 - User Table in Database

The home page of the SVP rendezvous host consists of multiple selections drop-down list box, start button, delete button, and add button as shown in figure 20. The multiple selections drop-down list box consist of IP addresses of the security managers associated with the compute hosts. The start button is used for starting the machines and the delete button is used for deleting the machine's IP addresses. The add button is used for adding new machines IP addresses.

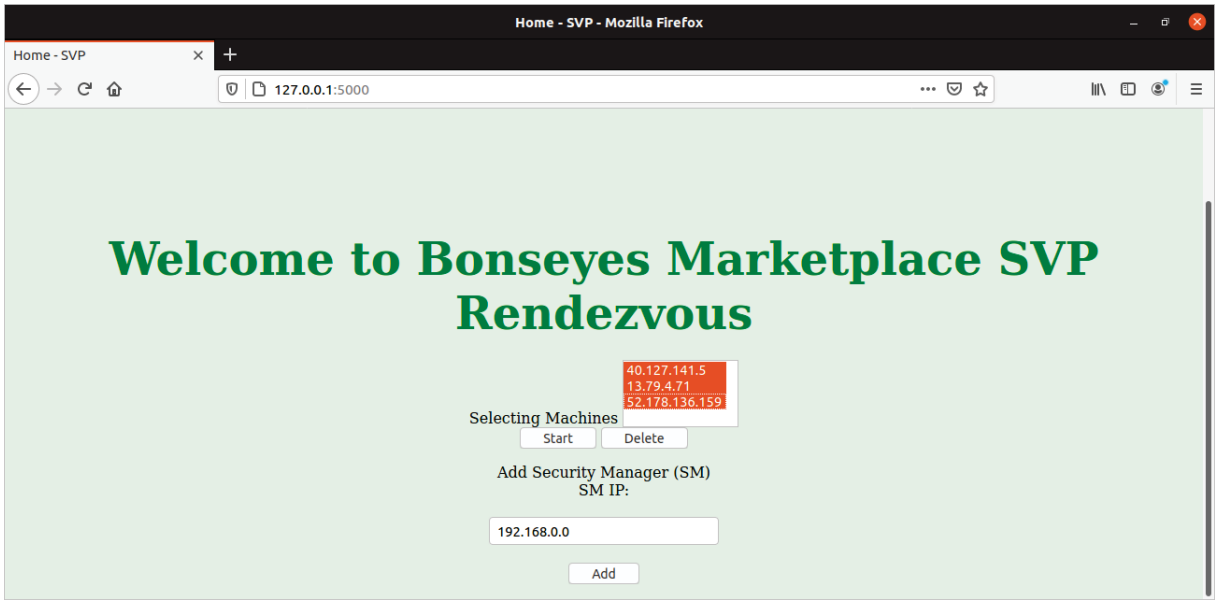

Figure 20 - Bonseyes Marketplace SVP Rendezvous Host

When a user selects the three machines and presses the start button, a query is sent to the SQLite database where IP addresses of the security managers associated with the compute hosts are stored as shown in figure 21. The data in the database is stored in the security manager (SM) table. The data is stored as rows and the data fields are id, smip, and user_id. The database used here is a relational database which means a user can have many security managers but each security manager will only have one user associated with it. The three ones in the user_id column in the database show the association of the security manager's IP addresses with the user.

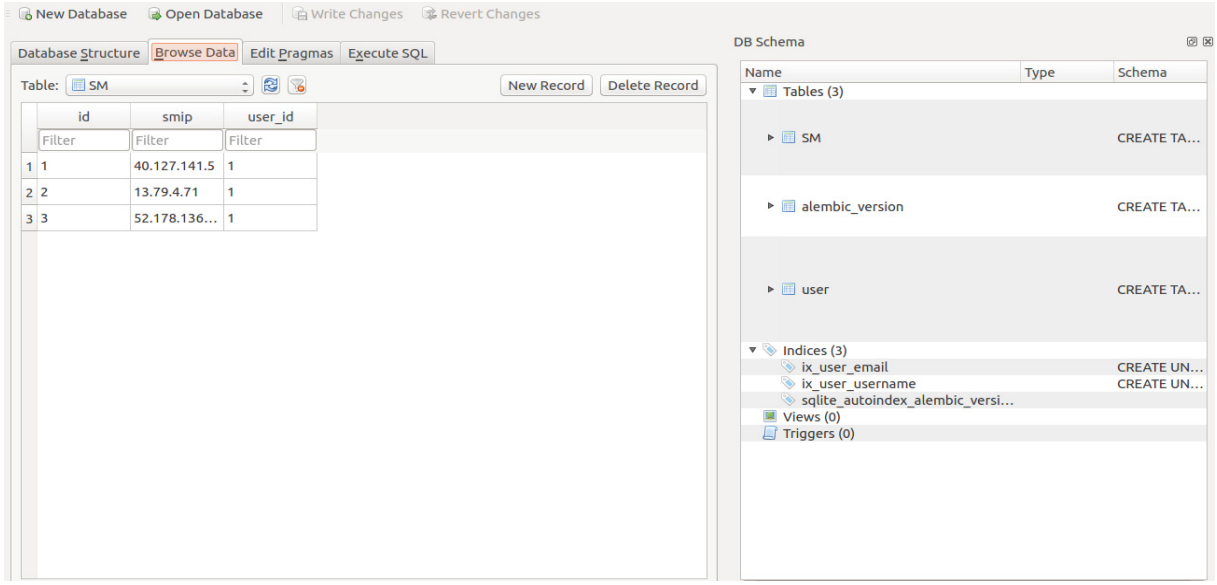

Figure 21 - Security Manager (SM) Table in Database

The user is then redirected to the security manager's webpages but the first user is requested for the user certificate verification. The certificates verification step is shown in figure 22.

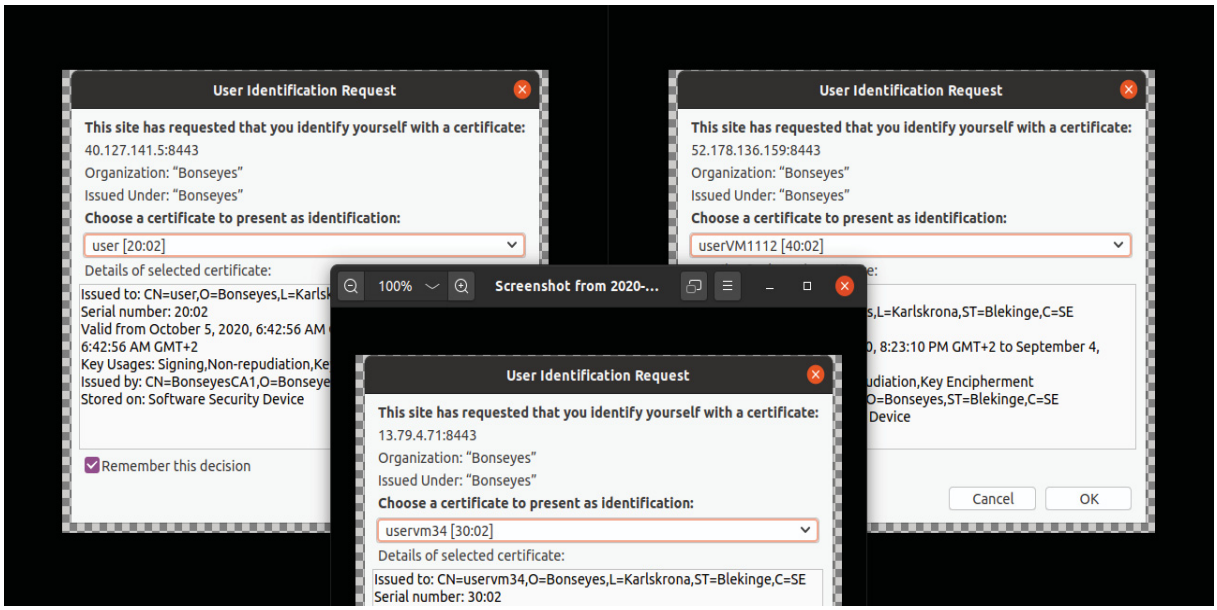

Figure 22 - User identify verification for accessing Multiple Security managers

Each certificate includes information about the owner's identity, serial number, validity, issuing authority, etc. The user certificates are installed in the web browser and the user chooses a certificate to present as identification and is redirected to the individual security managers. The webpages for the security managers are shown in figure 23.

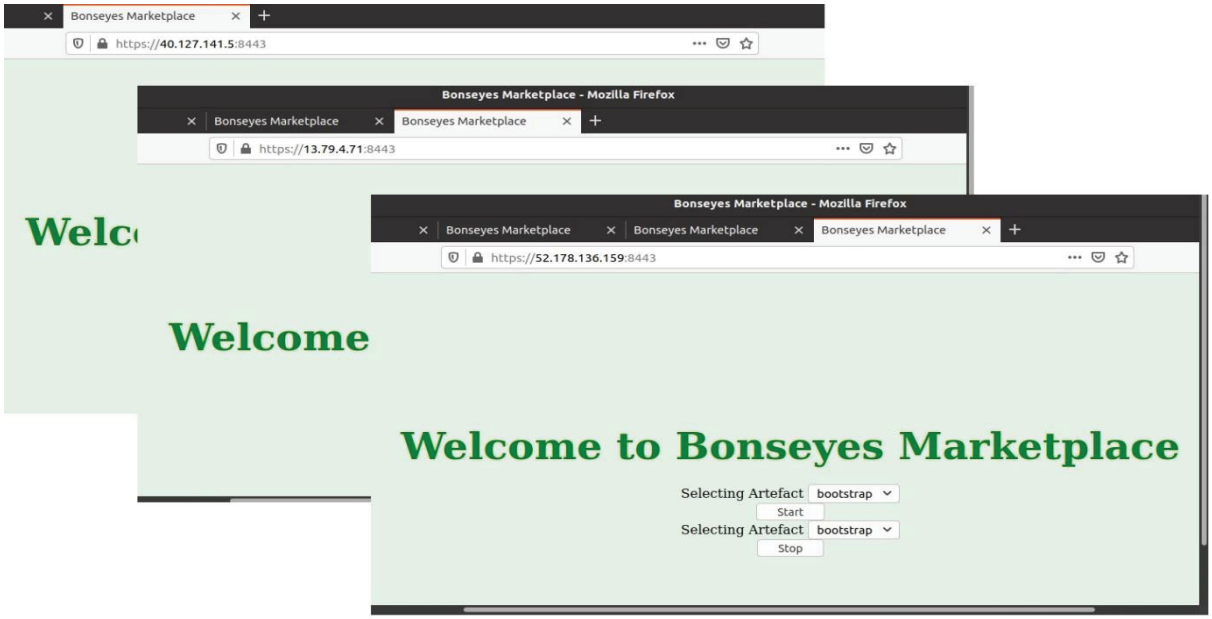

Figure 23 - Multiple Security Managers

The security managers consist of drop-down lists for selecting artifacts, a start button for starting the artifacts, and a stop button for stopping the artifacts. Now when the user presses the start button on each window to start the artifacts, the security managers contact the Bonseyes modules and download the bootstrap Docker images. Individual Security managers verify the integrity of the bootstrap Docker images and if the images are correct, the bootstraps start the Bonseyes layers. User identification is requested again for accessing the Bonseyes layers as shown in figure 24.

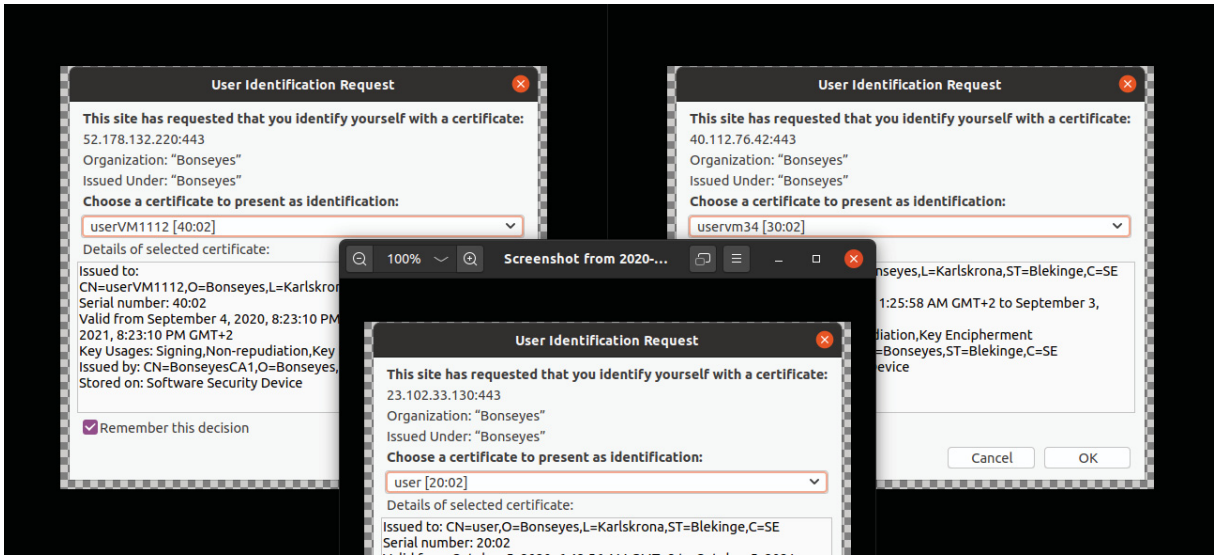

Figure 24 - User Identity Verification for accessing Bonseyes Layers

When the Bonseyes layers are started, the user is redirected to the Bonseyes layers webpage as shown in figure 25. Bonseyes layers provided information on the screen to proceed with the command-line interface using the destination IP address to manipulate various artifacts embedded in the AI pipelines.

Figure 25 - Multiple Bonseyes Layers started

5.6 Performance Evaluation

In this section, the results of the performance of our implementation are presented and analyzed by the start-up time of multi-location SVP. We verified the performance under a simplified scalability test. For statistical performance analysis, we initiated one, two, and three locations within one multi-location SVP for configuring a simple AI pipeline 1 as shown in figure 4, **Chapter 2**, **Section 2.1.1,** and measured the average start-up time.

We considered three scenarios, a) a user starting one location for executing a single AI tool, b) a user starting two locations for executing two AI tools, and c) a user starting three locations for executing three AI tools within a multi-side SVP. During the evaluation, the Marketplace rendezvous host was deployed on the local computer and each of the compute hosts was deployed on the Microsoft Azure virtual machines. For statistical performance analysis, we used the selenium framework to automate our solution by writing a test script in python programming language and measured the start-up time for each of these scenarios. Selenium is an open-source and free framework for automated testing of web applications. It uses a selenium web driver for accepting and sending commands to web browsers. We automated every task in the browsers as if a real user was executing the tasks. Time is measured as a time difference between the final time and start time. Each scenario is repeated 5 times and there is a 2 minutes gap between each iteration. Before evaluation, we stopped all the background applications to get accurate measurements. In the first scenario, we started one compute host to measure the time until the Bonseyes layer is set up and the user is notified that the orchestrator is ready. The time measured includes the certificate verification and license keys upload time for the AI artifact. The compute host was then stopped after repeating the experiment 5 times and we waited for 5 minutes to carry out another experiment. In the second scenario, two compute hosts were started and time was measured. In the last scenario, three compute hosts were started up and time was measured. Again the total time measured includes the certificate verification and license keys upload time for AI artifacts in all the 3 scenarios.

The results are present in Table [1]. The results are analyzed by comparing the different number of locations started in each scenario. It is represented in the graphical form below. From the experiments which we have carried out, it is clear that increasing or changing the number of compute hosts has very little impact on the start-up time.

Number of Compute Hosts	Start-up Time (Sec)	
	Mean (sec)	Standard Deviation
1	18.84	0.96
2	37.08	1.13
3	47.47	1.07

Table 1 - Mean, Standard Deviation of Start-up Time of One, Two, and Three Compute Hosts

The trend in figure 26 shows that as the number of compute hosts increases, the start-up time also increases. The x-axis represents the number of compute hosts and the y-axis represents the time until all the execution environments running.

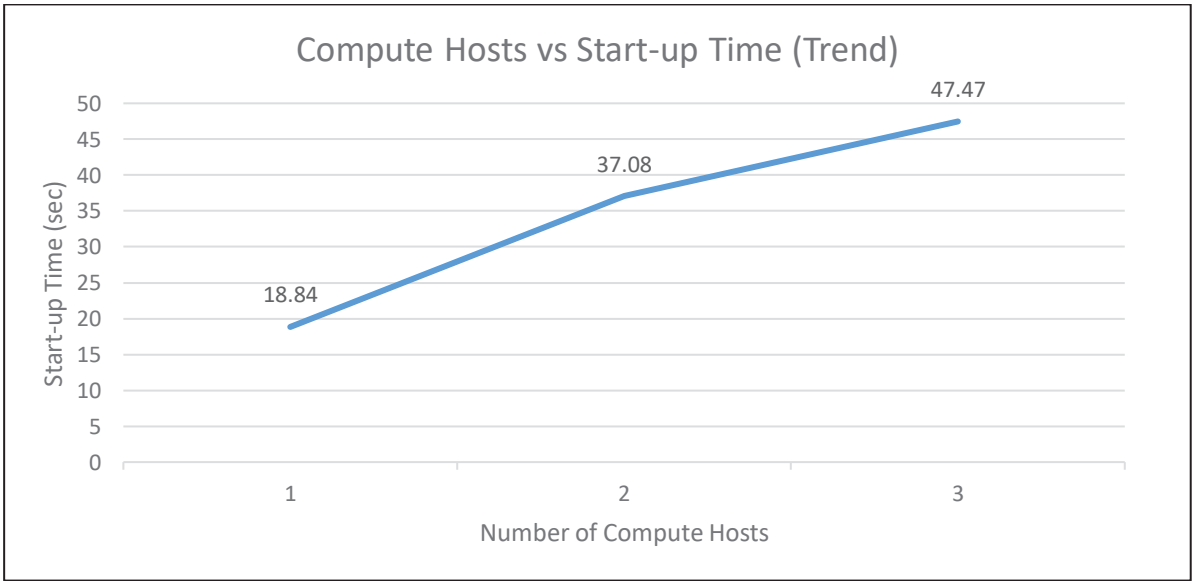

Figure 26 - Compute Hosts vs Start-up Time

Our experiment demonstrates how the core mechanism of starting the SVP work, that the whole system is useable and that the setup of a multilocation SVP can be done within a short time.

The time depends directly or linearly on the number of nodes. Hence, we believe that the multi-location SVP as outlined in this thesis enables the AI researcher to a) do the AI training in collaboration at different locations and b) to accelerate the AI application design.

6 DISCUSSION

6.1 Research Questions and Answers

RQ1: Which functions does one need to design and implement the federation of multi-location SVP?

- How can user-friendly configuration GUI for federated AI engineering be designed and implemented?

  The design and implementation of the GUI are based on the analysis of the user workflow. We abstractly identified the user's workflow, identified the major steps, and derived a mockup for the GUI for these steps and implemented it. The user's workflow includes configuration of the AI pipeline, placing AI artifacts on the remote compute resources, and starting a multi-location SVP for the execution of the AI pipeline.

- Which information need to be displayed in the GUI for distributed AI engineering?

  The GUI implemented for distributed AI engineering displays multiple selections drop-down list box, start button, delete button, and add button. The multiple selections drop-down list box consist of IP addresses of the security managers associated with the compute hosts of multi-location SVP. The start button is used for starting the machines and the delete button is used for deleting the machine's IP addresses. The add button is used for adding new machines IP addresses.

- Which functions are needed to start, control, and stop multi-location SVP?

  The implementation of starting multi-location SVP is done using for loop. When a user selects multiple machines and press the start button, start API is called, this API checks whether the user has access to the selected security managers and if this check passes, for loop starts and requests the IP addresses of the security managers associated with compute hosts stored in the SQLite database and opens them in a new window tab one by one. To add remote location IP address, a user enters the IP address, Add API is

called, add function checks the SQL database, if the IP address of the host is present it will flash host already added otherwise it will simply add a new entry to the database. To delete remote locations, a user selects the remote host from the list to delete, the loop starts and iterate over the IP addresses of security managers in the database and checks the validity of the request, if the request is not valid it will continue without performing any action otherwise it will simply delete the requested entry from the database. The stop function stops the remote location using the container name. It looks for the container name from the list of docker containers running and stops the container running against that name.

RQ2: How the data need to be structured for setting up multiple distributed locations?

The data of multi-location SVP is structured in the form of a list on the GUI. Whereas the data in the SQLite relational database is structured in the form of tables. We created one table for the users and the other table for the security manager's IP addresses. The user table is updated in the database when the User creates an account and registers itself on the Marketplace rendezvous host. The data in the database is stored as rows and consists of data fields like id, username, email address, and password hash. The security manager table is updated when the user Adds a new security manager IP Address. The data in the SQLite database is stored as rows and consists of data fields like id, smip, and user_id.

RQ3: How can these remote multiple locations securely be started-up and managed for the AI engineering task?

To start remote multiple locations securely, a user needs to install a user certificate in the web browser for authentication and authorization. Public key infrastructure technology is employed in each remote location. The communication between different entities of SVP is secured by server certificates and these certificates are not accessible to the users.

7 CONCLUSION AND FUTURE WORK

In this work, we researched the practical federation of resources into multi-location SVP and showing the feasibility of this concept by the implementation. The thesis transferred the concept of federation used in experimentation with networking experiments and transferred this towards the usability of AI Engineering.

This thesis dealt with the design, implementation, functional validation, and performance evaluation of the federated multi-location SVP for the collaborative development of AI models. A Graphical User Interface (GUI) is designed and out of this design requirements are derived for multi-location SVP. The multi-location SVP infrastructure consists of Marketplace Rendezvous Host. It is the centralized entity that governs the multiple independent distributed SVP compute resources. Marketplace rendezvous host hosts multiple security managers associated with the computing resources. These compute resources host the artifacts and are only accessible to the users who agree on the terms and conditions of its use with the artifact owner. Start, control, and stop functions are implemented on the Marketplace rendezvous host. A user can now start, control, and stop compute resources on demand. Users and compute resources related data is stored in the SQLite database. PKI is employed on each location for the authentication and authorization and HTTPs protocol is used for the secure communication between the entities of multi-side distributed SVP. The solution of multi-location SVP is structured in two parts: a) Functional validation and b) Numerical validation. Functional validation verifies the working of our implementation and Numerical validation verifies system useability and that the setup of a multi-location SVP can be done within a short time.

Future work: The multi-side distributed SVP provides an advantage for enabling multiple locations, more users, and multiple artifacts for trusted and secure collaborative AI development. This work can be enhanced by making it compatible and enable the interoperability of AI assets with the other AI marketplaces. This will increase the value of AI assets and will fasten the development time of AI application time.

8 REFERENCES

[1] T. Llewellynn *et al.*, "BONSEYES: Platform for Open Development of Systems of Artificial Intelligence: Invited paper," in *Proceedings of the Computing Frontiers Conference on ZZZ - CF'17*, Siena, Italy, 2017, pp. 299–304, doi: 10.1145/3075564.3076259.

[2] "BONSEYES – The Artificial Intelligence Marketplace," *BONSEYES*. https://www.bonseyes.eu/index.php (accessed Aug. 27, 2020).

[3] R.-V. Tkachuk, D. Ilie, and K. Tutschku, "Orchestrating Future Service Chains in the Next Generation of Clouds," p. 5.

[4] "OpenStack Docs: Overview." https://docs.openstack.org/install-guide/overview.html (accessed Aug. 27, 2020).

[5] "Production-Grade Container Orchestration," *Kubernetes*. https://kubernetes.io/ (accessed Aug. 27, 2020).

[6] M. de Prado *et al.*, "Bonseyes AI Pipeline -- bringing AI to you. End-to-end integration of data, algorithms and deployment tools," *ACM Trans. Internet Things*, vol. 1, no. 4, pp. 1–25, Aug. 2020, doi: 10.1145/3403572.

[7] M. Berman *et al.*, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, Mar. 2014, doi: 10.1016/j.bjp.2013.12.037.

[8] "GENI." https://www.geni.net/ (accessed Aug. 27, 2020).

[9] R. Ranjan and R. Buyya, "Decentralized Overlay for Federation of Enterprise Clouds," *arXiv:0811.2563 [cs]*, Nov. 2008, Accessed: Aug. 27, 2020. [Online]. Available: http://arxiv.org/abs/0811.2563.

[10] R. Riggio, T. Rasheed, and F. Granelli, "EmPOWER: A Testbed for Network Function Virtualization Research and Experimentation," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov. 2013, pp. 1–5, doi: 10.1109/SDN4FNS.2013.6702538.

[11] V. Ahmadi Mehri, "Towards Secure Collaborative AI Service Chains," 2019, Accessed: Aug. 27, 2020. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:bth-18531.

[12] "Cloud Computing Services | Microsoft Azure." https://azure.microsoft.com/en-us/ (accessed Aug. 27, 2020).

[13] V. Lozupone, "Analyze encryption and public key infrastructure (PKI)," *International Journal of Information Management*, vol. 38, no. 1, pp. 42–44, Feb. 2018, doi: 10.1016/j.ijinfomgt.2017.08.004.

[14] "What is PKI? | Entrust Datacard." https://www.entrustdatacard.com/pages/what-is-pki (accessed Aug. 27, 2020).

[15] M. J. Scheepers, "Virtualization and Containerization of Application Infrastructure : A Comparison," 2014. /paper/Virtualization-and-Containerization-of-Application-Scheepers/b06bc9d88762f5146445bd44e9a9deab174591dd (accessed Aug. 27, 2020).

[16] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs Containerization to Support PaaS," in *2014 IEEE International Conference on Cloud Engineering*, Mar. 2014, pp. 610–614, doi: 10.1109/IC2E.2014.41.

[17] "Empowering App Development for Developers | Docker." https://www.docker.com/ (accessed Aug. 27, 2020).

[18] Preeth E N, Fr. J. P. Mulerickal, B. Paul, and Y. Sastri, "Evaluation of Docker containers based on hardware utilization," in *2015 International Conference on Control Communication Computing India (ICCC)*, Nov. 2015, pp. 697–700, doi: 10.1109/ICCC.2015.7432984.

[19] Kurt Tutschku (Editor): "Deliverables D1.2 Revised Bonseyes System Architecture and Concepts", Technical Report by Bonseyes project, available on request at www.bonseyes.com, 2017

[20] Kurt Tutschku (Editor): "Deliverables D2.4 Revised Bonseyes System Architecture and Concepts", Technical Report by Bonseyes project, available on request at www.bonseyes.com, 2019


BLEKINGE TEKNISKA HÖGSKOLA
BTH