PrETTI2 Projektdokumentation

Architekturkonzept, Zugriffskonzept,
Installationsanleitung und Simulationsbericht.

gematik, IBM

11.07.2024

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument ist Teil der Liefergegenstände der IBM an die gematik im Zuge des Projekt-Auftrages PrETTI2. Bestandteil der Dokumentation sind das Architekturkonzept sowie das Zugriffskonzept. Neben der Dokumentation wird der gesamte Quellcode sowie Anleitungen zum Reproduzieren der Simulationsergebnisse beigelegt.

Dieses Dokument bündelt verschiedene Betrachtungsweisen auf das Gesamtsystem: Das Architekturkonzept beschreibt das generelle Design der Lösung sowie die architekturellen Abwägungen die im Verlauf des Projekts getroffen wurden. Das Zugriffskonzept stellt dar, wie die konzipierte Architektur die Datensicherheit gewährleistet in einer zukünftigen elektronischen Patientenakte ohne Ende-zu-Ende Verschlüsselung der Patientendaten. Der Abschnitt Installation beschreibt die notwendigen Schritte, um den Proof of Concept lokal oder in einem Test Cluster nachzuvollziehen. Im Abschnitt Simulation wird auf die Ergebnisse der Simulationsläufe eingegangen, welche im Test Cluster durchgeführt wurden.

1.1 Zielgruppe

Dieses Dokument richtet sich an ein technisch interessiertes Publikum, welches die Details der Lösung und die architekturellen Abwägungen besser verstehen möchte. Darüber hinaus bietet diese Dokumentation (zusammen mit dem bereit gestellten Quellcode) die Möglichkeit, den Demonstrator (in einer reduzierten Variante) selbst nachzubauen, zu modifizieren und zu evaluieren.

1.2 Fachlicher Hintergrund

Die Projektserie PrETTI hat sich zum Ziel gesetzt, die Machbarkeit und Skalierbarkeit von Privacy Enhancing Technologies (PETs) anhand konkreter medizinischer Informationssysteme in der Gesundheitsversorgung zu evaluieren. Mit PrETTI2 liegt der Fokus auf der Evaluierung von zwei ausgewählten PETs: Confidential Computing und Differential Privacy.

Orthogonal dazu wurde der fachliche Prozess der Forschungsdatenanfrage gewählt: über ein Portal soll es Forschenden ermöglicht werden, eine statistische Anfrage bezüglich der Gesundheitsdaten aller elektronischen Patientenakten (ePA) der Deutschen Patientenversorgung zu bekommen. Somit können potentiell 80 Mio. Versicherte angesprochen und in die Forschungsdatenerhebung einbezogen werden. Über die statistischen Anfragen hinaus sollen auch die Entwicklung und das Training von Modellen des Maschinellen Lernens unterstützt werden. So wurde der medizinische Anwendungsfall der Schlaganfall-Prävention ausgewählt: anhand von typischen Merkmalen sollen Patienten ein Risikofaktor dargestellt werden.

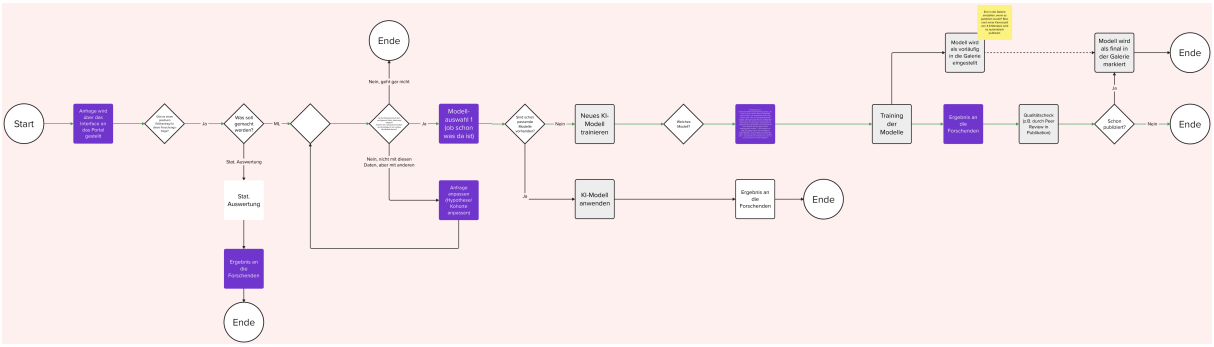


Abbildung 1: Fachlicher Prozessfluss

Abbildung 1 beschreibt die Abfolge einer Anfrage aus Sicht der Forschenden. Zunächst wird gewählt zwischen der Statistischen Auswertung und einem Modelltraining. Je nachdem, ob zu dieser Anfrage bereits ein trainiertes Modell vorliegt, wird dieses direkt angewendet, oder aber ein neues Modell zunächst trainiert. Nach Abschluss des Training wird das trainierte Modell in eine Galerie überführt, um es weiteren Forschenden zur Verfügung zu stellen.

1.3 Fachtermini

Im Folgenden sollen noch einige Grundbegriffe eingeführt werden, um die Verständlichkeit zu erhöhen und Uneindeutigkeiten vorzubeugen.

Ein **Forschungsauftrag** bezeichnet eine konkrete statistische Fragestellung (etwa “Wie viele Personen besitzen bereits die 3. COVID Impfung?”) oder eine konkrete Modell-Bildung (z.B. “Wie hoch ist mein Schlaganfall-Risiko, in Bezug auf Alter, Geschlecht, Ernährung und weiteren Faktoren?”). Zur Beantwortung gehört zunächst die Kohorten-Selektion (welche Patienten kommen überhaupt infrage?) sowie die Attribut-Auswahl (welche Datenpunkte sind relevant?).

Confidential Computing ist eine PET, die auf den Schutz von Daten bei der Nutzung (Data in Use) ausgerichtet ist. Die Technologie schützt Daten während der Verarbeitung, indem sie Berechnungen in einer hardware-gesicherten Enklave (Trusted Execution Environment, TEE) durchführt. Vertrauliche Daten werden erst dann für die Enklave freigegeben, wenn sie (die Enklave) als vertrauenswürdig eingestuft wurden.

Die **ePA-VAU** beschreibt solche eine sichere Enklave mit Zugriff auf die Patienten-Daten des Aktensystems. Sie ist eine konkrete Implementierung des Confidential Computing, und wird bereits in der elektronischen Patientenakte wie sie heute existiert eingesetzt.

Die **Forschungs-VAU** ist eine aktensystemübergreifende sichere Enklave für die Verarbeitung von Forschungsaufträgen. Es gibt nur eine übergreifende Instanz der Forschungs-VAU. Sie wird zentral durch eine öffentliche Einrichtung, etwa das BfArM, betrieben und den Forschenden zur Verfügung gestellt.

2 Architekturkonzept

Unter dem Leitgedanken “Bewege Fragen, nicht Daten” wird mit PrETTI2 der Prototyp einer Systemarchitektur vorgestellt, wie eine auf Privacy Enhancing Technologies basierende Verarbeitung von Gesundheitsdaten skalierbar realisiert werden kann. Dieses Architekturkonzept zeigt, wie ein zentraler Akteur einer Forschungsdatenplattform und mehrere ePA-Anbieter kooperieren, um gemeinsam statistische Analysen und Modelltrainings durchzuführen, ohne dass die sensiblen Patientendaten jemals die Aktensysteme verlassen. So zeigen wir auf, wie Daten der elektronischen Patientenakten datenschutzkonform und –freundlich mittels sogenannter “Privacy Enhancing Technologies” (PET) für die Primär- und Sekundärnutzung verarbeitet werden können. Ein weiteres Augenmerk liegt auf einer Skalierbarkeit der Lösung: das System soll in der Lage sein, mit den über 80 Millionen ePA Akten zu arbeiten und dennoch angemessene Reaktionszeiten zu bedienen.

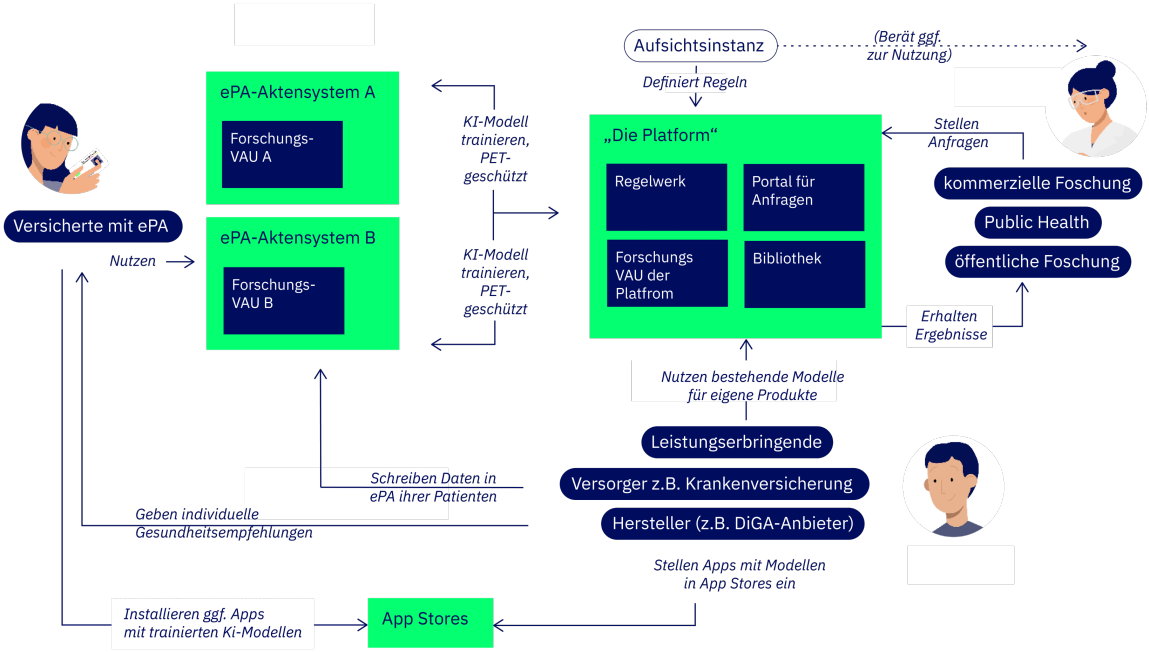


Abbildung 2: Gesamtkonzept, die involvierten Akteure sowie der resultierende Mehrwert-Kreislauf

Abbildung 2 visualisiert das Gesamtkonzept der Forschungsplattform im Zusammenspiel mit den bestehenden Aktensystem-Anbietern. Auf der linken Seite sind exemplarisch zwei ePA-Anbieter aufgeführt, welche den Zugriff auf die individuellen Akten der Versicherten verwalten. Auf der rechten Seite ist die Forschungsdatenplattform dargestellt, welche Anfragen der Forschenden entgegennimmt und zur Verarbeitung an die Aktensysteme weiter leitet.

Die versicherten Bürger (oben links) stellen über die Nutzung der ePA ihre Daten zur Verfügung und profitieren von individualisierten Empfehlungen sowie resultierende Gesundheitsanwendungen in Form von Apps (z.B. DiGAs) auf Basis der erstellten Modelle. Die Leistungserbringenden (unten rechts)

profitieren durch die Nutzung der Modelle in ihren eigenen Produkten, welche sie in Form von Gesundheitsmodellen und Apps an die Versicherten weiter geben. Die Forschenden (oben rechts) verarbeiten die eingestellten Daten, auf deren Basis sie ihre Forschungsarbeiten erstellen. Ebenfalls angedacht - in diesem Gesamtkonzept jedoch nur grob skizziert - ist die Einführung einer Aufsichtsinstanz, welche aufgrund von definierten Regeln die eingehenden Anfragen inhaltlich überprüft und missbräuchlicher Verwendung entgegen wirkt.

Das vorliegende Konzept skizziert eine Architektur, wie sie zukünftig möglich sein wird. Zum aktuellen Zeitpunkt verhindert die Ende-zu-Ende Verschlüsselung der Patientenakten die serverseitige Verarbeitung von Gesundheitsdaten. Die Aufhebung dieser ist jedoch im Zuge der ePA 3.0 (ePA für Alle) vorgesehen.

2.1 Fachliche Anwendungsfälle

Das Forschungsportal bringt die im vorherigen Abschnitt beschriebenen Akteure - Versicherte, Forschende und Leistungserbringer - zusammen und integriert sie in einen gemeinsamen Mehrwertkreislauf.

Dabei unterscheiden sich zwei Arten von fachlichen Anwendungsfällen:

1. **Statistische Anfragen** etwa zur Verteilung von demographischen Werten, Häufigkeiten von Krankheiten oder Immunisierungen. Diese werden in einem ersten Schritt vorbereitet (Kohortenauswahl) und und in einem zweiten Schritt beantwortet (Attribut-Auswahl).
2. **Anfragen zum maschinellen Lernen** ergänzen einen speziell zusammen gestelltem Datensatz (Kohortenauswahl und Attribut-Auswahl) um ein speziell gelerntes Machine Learning Modell (Modelltraining).

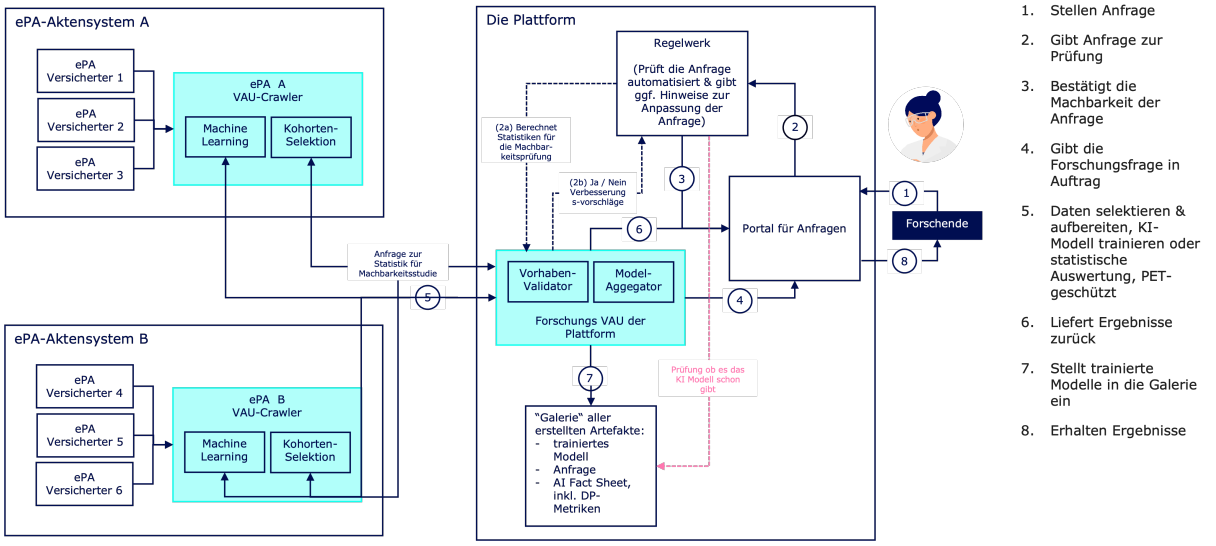


Abbildung 3: Fachlicher Flow einer Forschungsanfrage

Eine Visualisierung des fachlichen Flows einer solchen Anfrage wird in Abbildung 3 dargestellt. Nach Prüfung durch das Regelwerk und Bestätigung der Machbarkeit wird die Forschungsanfrage in Auftrag gegeben. Dabei leitet die zentrale Forschungsplattform die Anfrage an die einzelnen beteiligten Aktensysteme weiter und aggregiert die Ergebnisse.

Sowohl statistische Anfragen wie auch Modelltraining nutzen dabei das Multi Party Computing (MPC). Dadurch wird sicher gestellt, dass die Plattform selbst und somit auch die Forschenden zu keiner Zeit Zugriff auf die darunter liegenden Daten erhalten, sondern einzig allein aggregierte Statistiken oder (mit Differential Privacy verifizierte) Modelle erhalten. So wird eine Anfrage in Form eines JSON-Objektes an jedes beteiligte Aktensystem weitergeleitet, und nur die statistischen Ergebnisse bzw. die Modellupdates aggregiert an den Nutzer zurück gegeben.

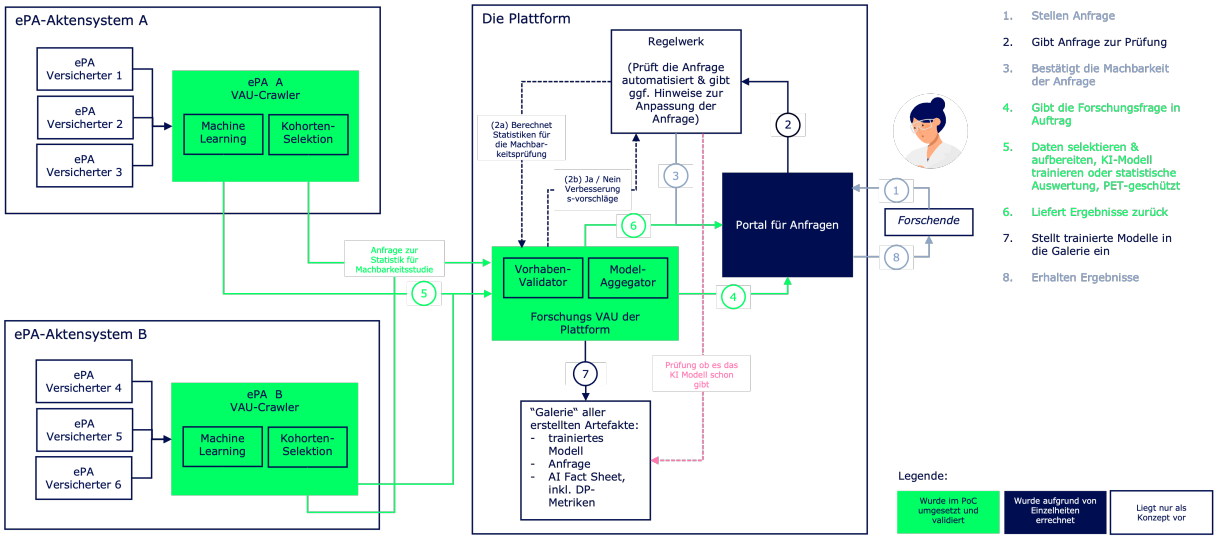


Abbildung 4: Abgrenzung von Gesamtkonzept zum Proof of Concept

Die in Abbildung 4 farblich hinterlegten Komponenten beschreiben die Komponenten, welche im Proof of Concept implementiert wurden. Das Regelwerk sowie die Modellbibliothek wurden ausschließlich konzeptionell angedacht. Das Portal für Anfragen liegt als Click Dummy vor, sowohl aus Sicht der Forschenden sowie der versicherten Bürger. Die folgenden fachlichen Komponenten - Kohortenbildung, Attribut-Auswahl und Föderiertes Lernen - wurden als Proof of Concept implementiert.

2.2 Fachliche Komponenten

Die Gesamtarchitektur kann in folgende fachliche Komponenten unterteilt werden:

- **Kohorten-Selektion**: Die initiale Eingrenzung der potentiell infrage kommenden Patienten, welches mittels Cuckoo Filter beschleunigt wird.

- **Attribut-Auswahl**: Die Bereitstellung eines aufbereiteten Datensatzes, welcher für statistische Abfragen oder Modell-Trainings verwendet werden kann.
- **Föderiertes Lernen**: Wird ein Modelltraining durchgeführt, so geschieht das mit Hilfe der PETs Federated Learning und Differential Privacy.

Auf die einzelnen Komponenten soll nun im Folgenden genauer eingegangen werden.

2.2.1 Kohorten-Selektion

Jede Forschungsanfrage kann und sollte Selektion-Kriterien enthalten, welche die Menge der zutreffenden Patienten eingrenzen. Fur eine initiale Evaluierung der Fragestellung ist es von Vorteil, möglichst schnell die Menge der potentiell zutreffenden Patienten zu erfahren - auch ohne die konkreten Ergebnisse der Abfrage zu erhalten.

Für die initiale Bewertung der Anfrage wird daher initial nur die Anzahl der zutreffenden Patienten zurück geliefert. Eine Beispiel Anfrage kann dabei so aussehen:

```
POST /select HTTP/1.1
Content-Type: application/json
Host: localhost:8083

["gender='male'", "birthDate < @1988"]
```

Als Payload wird eine Liste von Filterkriterien nach dem FhirPath[1] Standard übergeben.

In PrETTI2 werden Cuckoo Filter eingesetzt, um die Kohorten-Selektion zu beschleunigen (für eine genaue Erklärung siehe Abschnitt Cuckoo Filter). Die Menge der aller potentiellen Kandidaten einer Forschungsanfrage wird durch den Cuckoo Filter eingegrenzt auf die Menge derer, die mit hoher Wahrscheinlichkeit das Merkmal enthalten. Die reduzierte Menge muss dann jedoch erneut auf das Merkmal überprüft werden (eine zeitaufwändigere Operation) um die falsch-positiven Ergebnisse auszusortieren.

2.2.2 Attribut-Auswahl

Nach der Eingrenzung der potentiellen Kandidaten wird im Modul der Attribut-Auswahl die eigentliche Auswahl der Attribute vorgenommen. Eine Beispiel Abfrage kann mit folgendem Aufruf gestartet werden:

[1] https://build.fhir.org/fhirpath.html

```
POST /query HTTP/1.1
Content-Type: application/json
Host: localhost:8083

{
  "name": "Retrieve all female patients that were born between 1968 and
      1988. Show me the following attributes: Patient ID, Given Name,
     Family Name, Birth Date, Gender. Also include whether there they
     are older than 35.",
  "filters": ["gender='female'", "birthDate >= @1968", "birthDate <=
     @1988"],
  "columns": {
    "Patient ID": "id",
    "Given Name": "name.first().given.first()",
    "Family Name": "name.first().family",
    "Birth Date": "birthDate",
    "Gender": "gender",
    "Older than 35": "birthDate < @1988"
  }
}
```

Der Parameter `name` ist dabei nur als Beschreibung gedacht. Essentiell sind die Parameter `filters` (welcher analog zur Kohorten-Selektion die Menge der zu betrachtenden Akten eingrenzt) sowie `columns`: letzteres beschreibt die gewünschten Attribute und die FhirPath Query um diese zu selektieren. Dazu muss jede Akte einzeln eingelesen, geparsed und abgefragt werden. Je nach Art der Speicherung der Daten und resultierendem Netzwerkverkehr kann dies sehr lange dauern. Im folgenden Abschnitt soll daher eine Optimierung der Zugriffszeiten mittels Cuckoo Filter beschrieben werden.

2.2.3 Cuckoo Filter

Zur Reduzierung der Latenz bei der Abfrage werden die berechneten Werte mittels Cuckoo Filter[2] zwischengespeichert. Cuckoo Filter sind eine Weiterentwicklung der Bloom Filter, einer so genannten probabilistischen Datenstruktur: Sie gleichen in der Verwendung der Datenstruktur ‘Set’ (Menge), jedoch sind falsch-positive Zugriffs-Ergebnisse möglich.

Ein Bloom-Filter ist eine platzsparende probabilistische Datenstruktur, die es ermöglicht, zu überprüfen, ob ein Element **nicht** in einer Menge enthalten ist. Anders gesagt: ein Bloom-Filter kann uns entweder sagen, dass ein Element in einer Menge enthalten sein *könnte*, oder dass ein Element *definitiv nicht* in einer Menge *enthalten ist*. Der Hauptgrund für den Einsatz von Bloom-Filter ist ihr geringer Speicherverbrauch.

[2] https://en.wikipedia.org/wiki/Cuckoo_filter

Um einen Bloom-Filter zu befüllen, beginnt man mit einem Null-Bit-Array. Jedes Mal, wenn ein Element in den Bloom-Filter einfügt werden soll, wird zunächst der Wert des Elements k-mal gehasht, und die resultierenden Indizes auf die die Hashes zeigen, werden auf den Wert '1' gesetzt. Um zu prüfen, ob ein Element im Bloom-Filter vorhanden ist, prüfen wir einfach, ob nach dem Hashing alle resultierenden Indizes den Wert '1' enthalten. Wenn einer der Indizes immer noch eine '0' ist, wissen wir, dass das Element nie in den Bloom-Filter eingefügt wurde. Der falsch-positive Charakter des Bloom-Filters ergibt sich aus der Tatsache, dass die Werte einige Indizes für ein bestimmtes Element bereits auf '1' stehen, obwohl wir das Element nie eingefügt haben.

2.2.4 Föderiertes Lernen

Beim maschinellen Lernen wird ein lokales Training auf dem Aktensystem durchgeführt, dessen Ergebnisse über Federated Learning an die Forschungsdatenplattform weitergeleitet werden. Dazu senden die einzelnen FL Agenten - welche in den Aktensystemen verortet sind - regelmäßig ihre aktualisierten Modellparameter an den zentralisierten FL Server, welcher in der Forschungsplattform lokalisiert ist. Somit wird sicher gestellt, dass diejenigen, die mit den Daten arbeiten wollen (z.B. Forscher oder die Öffentliche Gesundheit), die Daten selbst niemals erhalten. Die Daten bleiben immer im ePA-Aktensystem und verlassen dieses nie. Stattdessen werden die Fragen zu den Daten übermittelt, bei den Daten (im ePA-Aktensystem) beantwortet und nur die Antworten gelangen zurück zum Fragesteller.

2.3 Datenaufbereitung

Ein wichtiger Aspekt der Clients ist das Abrufen der Datenauswahl aus den ePA-Datensätzen. Um die Interaktion mit einem elektronischen Krankenaktensystem zu simulieren, wurden synthetische Ressourcenbündel mit Synthea[3] erstellt. Diese werden als eine Sammlung von Dateien gespeichert. Jeder Client holt eine partitionierte Menge davon ab, lädt sie in den Speicher und führt die Attributauswahl mit `fhirpath` durch. Die Attribute werden dann in einem Pandas DataFrame zusammengefasst und für das Federated Learning gespeichert.

2.4 Gramine Konvertierung

Zur Verschlüsselung des Hauptspeichers via Intel SGX sowie der Remote Attestation wird Gramine[4] verwendet. Die dazu benötigten Konfigurationsdateien befinden sich in dem Unterordner `gramine`. Zur Konvertierung des Services wird ein Makefile bereitgestellt, welches unter `gramine/Makefile`

[3] https://github.com/synthetichealth/synthea
[4] https://gramineproject.io/

zu finden ist. Mit einem selbst erstellen Signing Key wird das Service Image signiert und ein SGX-spezifisches Manifest (`tensorflow_model_server.manifest.sgx`) erzeugt.

Führen Sie `make DEBUG=1` im Ordner `gramine` aus, um die Gramine-bezogenen Dateien zu erstellen. Verwenden Sie `make clean`, um sie zu entfernen. Die Datei `tensorflow_model_server.manifest.template` sollte generisch genug sein, um so zu bleiben wie sie ist.

Das `start.sh` Skript, baut eine Verbindung zum PCCS Service auf und startet dann den Tensorflow Model Server in der SGX Enklave.

3 Zugriffskonzept

Das folgende Kapitel betrachtet die Gesamt-Architektur in Bezug auf Datensicherheit und Zugriffsberechtigungen. Umfang ist die Beantwortung der Frage “wie werden Daten einer Forschungs-VAU zur Verfügung gestellt?” Anhand von Sequenzdiagrammen wird dargestellt, wie die schützenswerten Daten im ePA-Aktensystem vor unberechtigtem Zugriff abgesichert werden. Der Fokus liegt dabei auf der lokalen Datenverarbeitung: keine Patientendaten verlassen die Enklave - einzig aggregierte Statistiken sowie (verrauschte) Modell-Parameter werden an die Forschungsdatenplattform übermittelt.

Wie wird hierbei sicher gestellt, dass die Daten vor unberechtigtem Zugriff geschützt sind? Grundsätzlich lassen sich dabei drei Arten der Datennutzung unterscheiden, welche separate Verschlüsselungen benutzen:

1. Data at Rest: Persistierte Daten, welche in den einzelnen ePA Akten gespeichert liegen. Diese werden über den Document Key verschlüsselt gespeichert.
2. Data in Transit: Daten, welche geladen und in den VAU-Crawler transportiert werden. Mittels Transport Layer Security (TLS) wird dieser Transport abgesichert.
3. Data in Use: Daten, welche im Hauptspeicher des VAU-Crawler verarbeitet werden. Mittels der Intel SGX Technologie wird der Hauptspeicher verschlüsselt und der Zugriff gesichert.

Der Datenschutz wird darüber hinaus durch eine Kombination von Federated Learning und Differential Privacy gewährleistet. Konkret wurde dies durch die Maxime „bewege Fragen, keine Daten“ wie folgt umgesetzt: Die VAU der Aktensysteme ist der einzige Ort, an dem Patientendaten im Klartet verarbeitet werden. Durch technische Maßnahmen wie die Verwendung von Enklaven (Verschlüsselung des Hauptspeichers, Remote Attestation) ist dieser geschützte Raum vor Zugriffen von außen abgesichert - nicht einmal der Betreiber der Plattform bekommt Zugang zu den sensiblen Patientendaten. Durch Federated Learning kann auch bei Aktensystem übergreifenden Berechnungen die Datenverarbeitung lokal im ePA Aktensystem stattfinden. Durch Differential Privacy wird sicher gestellt, dass aus den Ergebnissen keine Rückschlüsse auf individuelle Datenpunkte hergestellt werden kann.

Informationen, welche die VAU verlassen, sind stets nur aggregierte Statistiken oder verrauschte Modellparameter. Auch die forschende Person bekommt somit keine Einsicht in die einzelnen Datenpunkte, auf welchen die Beantwortung der Forschungsfrage basiert.

Mit der Einführung der ePA für Alle (ePA 3.0) wird es keine Ende-Zu-Ende Verschlüsselung der Patientenakten mehr geben. Die Sicherheitsarchitektur der ePA 3.0 sieht vor, dass statt einer Ende zu Ende Verschlüsselung der Aktenkonten diese im Klartext gespeichert, jedoch durch ein Berechtigungskonzept (den so genannten Befugnissen) autorisiert werden.

Als Resultat der mit ePA 3.0 angepassten Datenspeicherung fällt der record key weg. Somit gibt es keine Abhängigkeit zum FdV mehr, und ein VAU-Crawler braucht stattdessen nur eine entsprechende Befugnis auf die Daten, die auch zur Forschung freigegeben wurden. Die Granularität dieser Befugnis kann dann über so genannte Policies weiter eingeschränkt werden. Auch hier greift wieder der Leitgedanke "Bewege Fragen, nicht Daten": die eigentlichen Daten verlassen nie den abgesicherten Raum der Enklave. Stattdessen werden die Anfragen direkt in den VAUs bearbeitet, und nur die aggregierten Ergebnisse werden zurück übermittelt.

3.1 Annahmen und Vorbedingungen

Vorab sollen einige Annahmen deutlich gemacht werden, unter welchen Rahmenbedingungen eine zukünftige ePA mit Forschungsdaten agieren kann:

1. Es wird je Aktensystem-Anbieter ein so genannter VAU-Crawler entwickelt: diese vertrauenswürdige Ausführungsumgebung bekommt lesenden Zugriff auf die Aktenkonten mittels einer entsprechenden Befugnis, so dass der Crawler unabhängig agieren kann.
2. Aufgrund der existierenden Befugnisse werden auch neu eingestellte Dokumente direkt für den VAU-Crawler freigegeben.
3. Bei der Bearbeitung eines neuen Forschungsauftrags wird (nach erfolgreichem Durchlaufen der Vorabprüfung) der VAU-Crawler beauftragt, sämtliche ePA-Akten zu durchsuchen, und bei Erfüllung der Kriterien das definierte Subset an Attributen in einen separaten Daten-Frame zu kopieren.
4. Für statistische Anfragen zum Forschungsauftrag wird dieser Daten-Frame aggregiert und die Ergebnisse an die Forschungs-Plattform übermittelt..
5. Das Modell-Training via Federated Learning geschieht als Kooperation der Forschungs-VAU mit den VAU-Crawlern der Aktensystem-Anbieter, welche auf ihren anbieterspezifischen Daten-Frames trainieren und nur die Model-Updates weiterreichen.
6. Für die Inferenz wird das trainierte Modell in einer Forschungs-VAU bereitgestellt.

Zusammengefasst beschreibt jeder Forschungsauftrag somit zunächst einen auf die Forschungsfrage angepassten Daten-Frame, welcher dann wahlweise für statistische Abfragen zu Resultat-Werten oder für das Modell-Training zu einem trainierten Modell weiter verarbeitet wird.

3.2 Initialisierung

Auch das neue ePA 3.0 Aktensystem muss mit bestehenden Aktenkonten kompatibel agieren. Daher ist es nötig, bestehende ePA Akten für die Forschungsfreigabe zu initialisieren. Aufgrund der mit ePA 3.0 eingeführten Befugnisse wird dies einfach und konform der bestehenden Architektur ermöglicht.

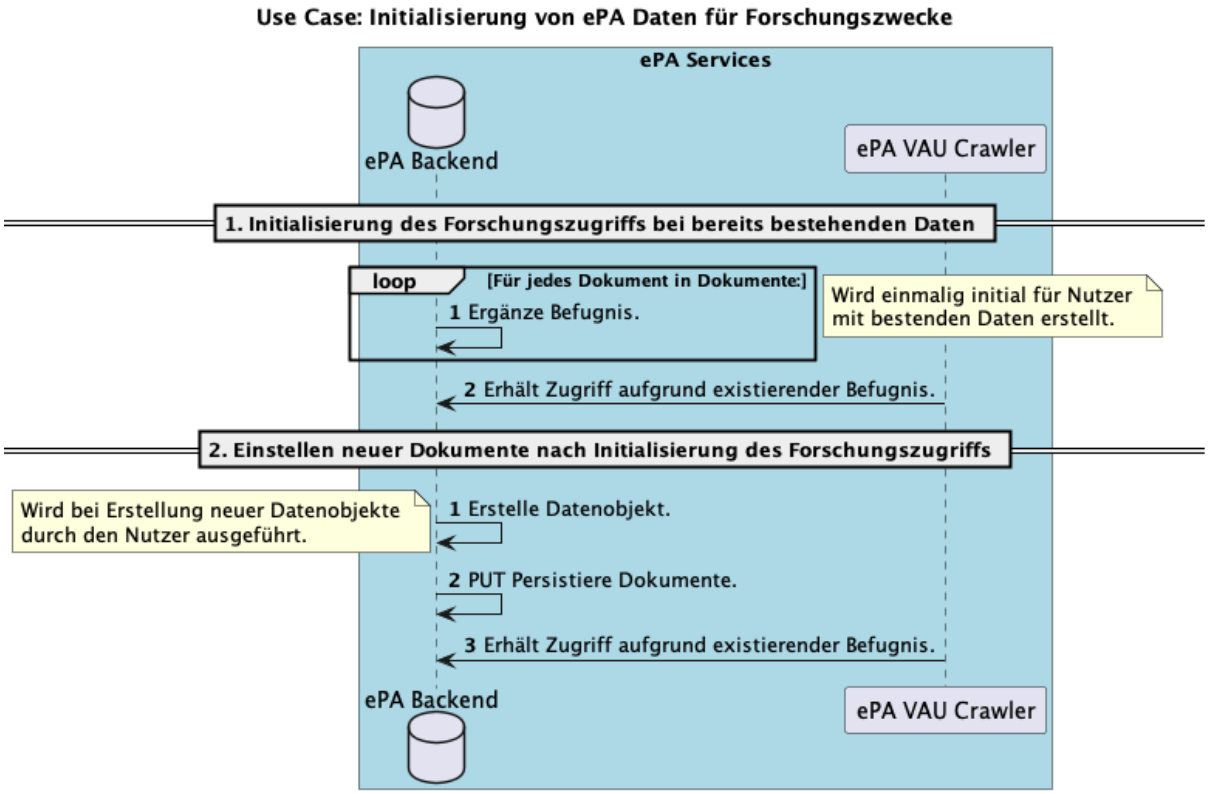


Abbildung 5: Initiale Freigabe von ePA Daten

Abbildung 5 beschreibt die Initialisierung einer bereits existierenden Patientenakte für den Zugriff durch den VAU-Crawler. Durch Erteilung der Befugnisse durch den Patienten wird der VAU-Crawler befähigt, die Daten zu lesen. Beim Einstellen neuer Dokumente greifen die bestehenden Befugnisse, so dass der VAU Crawler direkt Zugriff erhält. Durch die Opt-Out Regelung können diese Freigaben jederzeit granular eingestellt oder widerrufen werden. Somit wird über die bereits durch die ePA 3.0 spezifizierten Befugnisse sicher gestellt, dass nur berechtigte Akteure Zugriff auf die Daten bekommen.

3.3 Statistiken

Sind die Daten für die Forschungsanfrage frei gegeben, so können diese für die Berechnung von statistischen Anfragen sowie Modelltrainings weiter verarbeitet werden.

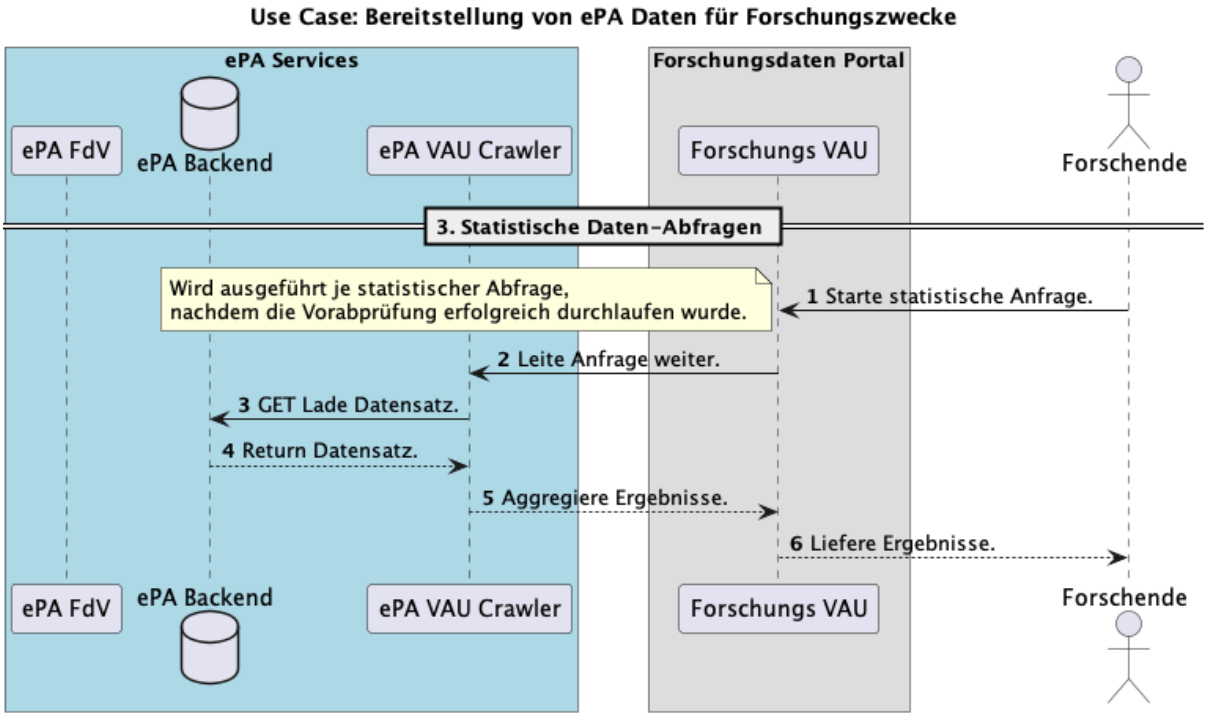


Abbildung 6: Statistische Daten-Abfragen

Wie in Abbildung 6 dargestellt, beginnt die Anfrage nach Statistiken im Forschungsportal, und wo zunächst die Vorabprüfung durchgeführt wird. Nach erfolgreicher Vorabprüfung wird für die Beantwortung der statischen Daten-Abfragen die Anfrage zunächst von der Forschungs-VAU an die jeweiligen ePA VAU Crawler der Aktensystem-Anbieter weiter geleitet. Die VAU-Crawler laden daraufhin den Datensatz des jeweiligen Aktensystems, selektieren und aggregieren die gewünschten Statistiken und leiten die Ergebnisse zurück an die Forschungs-VAU. Die Forschungs-VAU wiederum aggregiert die Ergebnisse und liefert sie aufbereitet an den Forschenden zurück. Da nur aggregierte Statistiken die VAU-Crawler verlassen, wird sicher gestellt, dass keine sensiblen Personendaten den schützenswerten Raum der VAU verlassen.

3.4 Modelltraining

Das maschinelle Lernen von Modellen passiert wie folgt: Die Anfrage wird zunächst asynchron an die einzelnen Aktensysteme weiter geleitet. Jedes Aktensystem führt mittels VAU-Crawlern parallelisiert das Laden der Dokumente sowie die Extraktion der Attribute durch. Aufgrund der in Abbildung 1 dargestellten initialen Befugnis zum Lesen der Dokumente kann der VAU Crawler die freigegebenen Dokumente beziehen und auswerten. Die Ergebnismengen werden aggregiert und der resultierende Datensatz wieder im ePA Backend persistiert.

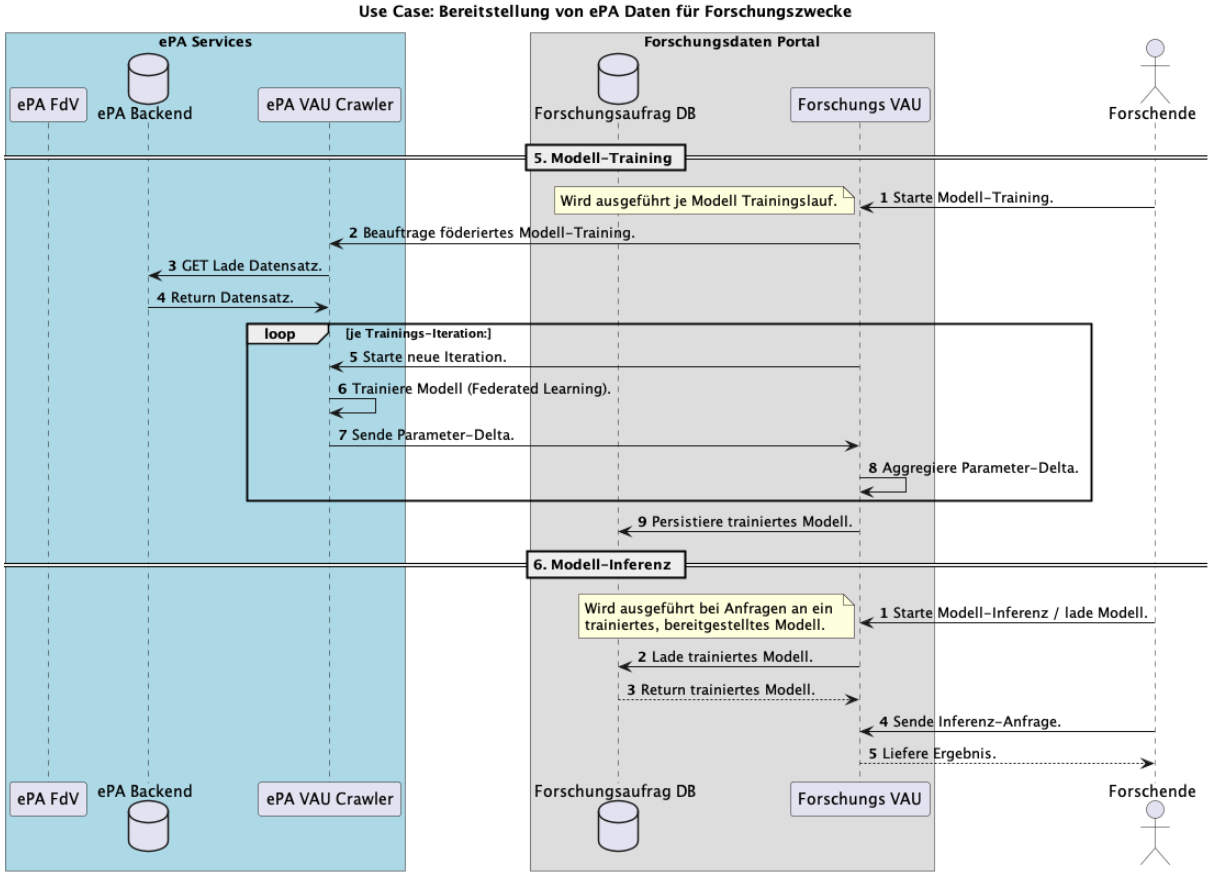


Abbildung 7: Modelltraining mittels Föderiertes Lernen

Ausgehend von einem Auftrag zum Föderierten Lernen seitens der Forschungs-VAU werden die VAU-Crawler der Aktensysteme beauftragt, den in Abbildung 7 bereit gestellten Datensatz zu laden und individuell das Modelltraining zu starten. Je Trainingsrunde ‘round’ werden die Datensätze ‘batch_size’ Mal durchlaufen, und nach jeder Runde werden die gelernten Model-Parameter zur Aktualisierung an den FL-Server in der Forschungs-VAU geschickt. Der Durchschnitt der gelernten Parameter wird daraufhin an alle beteiligten Clients - die VAU-Crawler - übermittelt und die nächste Trainingsrunde beginnt. Nach Abschluss des Trainings wird das gelernte Modell persistiert und für die spätere Weiterverwendung hinterlegt. Zur Modell Inferenz wird das trainierte Modell aus der Galerie geladen und über einen Endpunkt zur Verfügung gestellt.

4 Simulation im Test Cluster

Im folgenden Kapitel wird beschrieben, wie mit Hilfe von synthetisch generierter Patientendaten exemplarische Anfragen simuliert und das Laufzeitverhalten und die Skalierbarkeit des Systems untersucht wurden.

4.1 Technischer Versuchsaufbau

In diesem Abschnitt soll genauer auf den technologischen Unterbau und die verwendeten Werkzeuge, Bibliotheken und Hilfsmittel eingegangen werden.

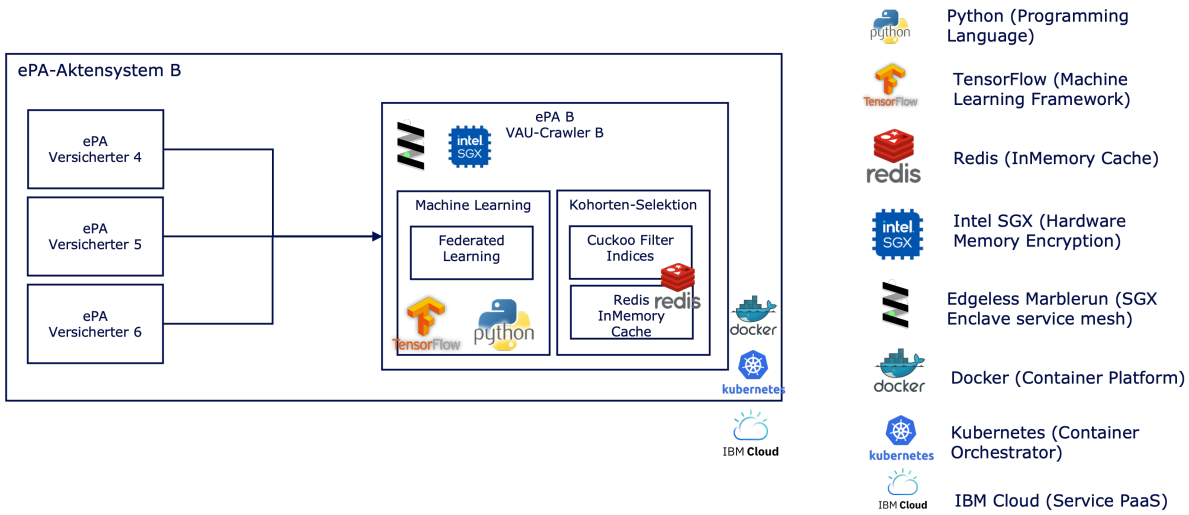


Abbildung 8: Technologie Stack der Implementierung des Demonstrators

Das im Architekturkonzept beschriebene System wurde mit den folgenden Technologien als Demonstrator umgesetzt:

- **Python**: Python ist eine dynamisch typisierte Programmiersprache und vor allem verbreitet im Einsatz für Wissenschaft und Forschung, Data Science, Machine Learning, Webentwicklung und Automatisierung. Python ist bekannt für seine simple und lesbare Syntax, die es ermöglicht, schnell und effizient Lösungen zu entwickeln. Die Sprache unterstützt verschiedene Paradigmen, wie z.B. Prozedurale, Objektorientierte und Funktionale Programmierung.
- **Redis/ValKey**: Ein InMemory Cache, welcher optimierte Datenstrukturen im Hauptspeicher bereit stellt. Redis wird vor allem für die Implementierung der Cuckoo Filter in der Kohorten-Selektion verwendet, um die Zugriffszeiten auf gespeicherte Abfragen zu beschleunigen.
- **Docker**: Docker ist eine Plattform, die es erlaubt, Anwendungen in isolierten Umgebungen (so genannten Containern) zu betreiben. Jeder Container ist ein eigenes, isoliertes System mit eigener Betriebssystemumgebung und zugewiesenen Ressourcen (z.B. RAM, CPU). Da jeder

Container nur den benötigten Prozess und seine Abhängigkeiten enthält, können Anwendungen und Services in einer effizienten, skalierbaren und sicheren Weise ausgeführt werden.
- **Kubernetes**: Kubernetes ist ein Container-Orchestration-System, das es ermöglicht, mehrere Docker-Container in einem Cluster zu verwalten und auszuführen.
- **Helm**: Helm ist ein Package Manager für Kubernetes, ähnlich wie apt (Advanced Package Tool) oder pip für Linux-Distributionen oder Python-Packages. Helm ermöglicht es, Kubernetes-Anwendungen und -Tools zu installieren, zu aktualisieren und zu verwalten.

Die fachlichen Module selbst wurden mit Python unter der Verwendung der Machine Learning Bibliothek Tensorflow[5] entwickelt. Der Demonstrator Code wurde in zwei Module als Docker Container getrennt designt: Zum Einen die Plattform, welche das Anfrage-Interface sowie den Federated Learning Modell-Aggregator enthält, zum Anderen den VAU-Crawler, bestehend aus Kohorten-Selektion und Federated Learning Agent. Jedes der Module wurde mittels eines Dockerfile als Container und via Helm als Paket bereit gestellt.

Im Folgenden soll nun auf die konkreten Hardware Spezifikationen der Testumgebung eingegangen werden.

4.2 Infrastruktur

Die Evaluierung des Demonstrators wurde auf einem Test Cluster der IBM Gesundheitsplattform im Rechenzentrum Frankfurt FRA02 der IBM Cloud durchgeführt: Zwei Worker Nodes vom Prozessortyp `Intel SGX amd64 me4c.4x32.encrypted` wurden mit `Ubuntu 20.04.6 LTS`, `Kubernetes 1.28.4_1541` und `Marblerun Enterprise 1.3.0` in einem privaten VLAN bereit gestellt. Beide Knoten verfügen über jeweils 4 Prozessor-Kerne, `32GB` Arbeitsspeicher, `128MB` Enclave Page Cache sowie eine maximale Netzwerkgeschwindigkeit von `10Gbps`. Drei Kubernetes Pods wurden dann via Marblerun auf den Cluster ausgerollt: eine Instanz der Plattform sowie zwei Instanzen des VAU-Crawlers.

Zur Evaluierung des Laufzeit-Verhaltens und der Skalierbarkeit wurden die Kohorten-Selektion sowie das Modelltraining mehrfach auf dem Test Cluster ausgeführt. Die Bestimmung des Laufzeitverhaltens wurde über Performance Logging mit Messpunkten am Anfrage-Interface der Plattform sowie dem Federated Learning Modell-Aggregator durchgeführt.

4.3 Synthetische Datengenerierung von FHIR Ressourcen

Zur Evaluation der Architektur in Bezug auf Skalierbarkeit und Performance wurden Messungen mit Beispieldaten durchgeführt. Fur die realitätsnahe Simulation der elektronischen Patientenakten wurden

[5] https://www.tensorflow.org/

Patientendaten mit dem frei verfügbaren “Synthetic Patient Population Simulator Synthea[6] erstellt. Synthea ist ein ein Open-Source-Generator zur Modellierung der Krankengeschichte synthetischer Patienten. Der Unterordner `/synthea` der Quellcode Verzeichnis enthält die entsprechende Konfiguration sowie eine Anleitung zur Erstellung von Beispiel Datensätzen. Das Programm Synthea selbst kann über das Quellcode Repository[7] herunter geladen werden. Darüber hinaus sind im Ordner `data` einige bereits generierte Beispiele zu finden, welche direkt lokal in der Simulation verwendet werden können.

Die folgende Konfigurationsdatei `config.properties` beschreibt die Parameter zur Erzeugung von 100 Beispielpatienten.

```
generate.default_population = 100
exporter.use_uuid_filenames = true
exporter.fhir.bulk_data = true
exporter.fhir.transaction_bundle = false
exporter.years_of_history = 1
exporter.fhir.export = true
exporter.text.export = false
exporter.hospital.fhir.export = false
exporter.practitioner.fhir.export = false
```

Die Generierung der Testdaten erfolgt über den folgenden Befehl:

```
java -jar synthea-with-dependencies.jar -c custom.properties
```

Die erzeugte Datei enthält je Zeile eine Akte, welche im Anschluss für die effiziente Weiterverwertung in das Protocol Buffer Format konvertiert werden können. Dazu führt man folgenden Befehl aus:

```
python cohort/protobuf.py
```

4.4 Kohorten-Selektion

Zur Evaluierung der Skalierbarkeit der Kohorten-Selektion wurden 2 Mio. Beispiel Datensätze erzeugt, welche dann auf den IBM Test Cluster weiter verarbeitet wurden.

Dabei wurden zunächst 2 Millionen Beispielakten mit dem in Abschnitt 4.2 beschriebenen Prozess erstellt. Diese Gesamtmenge wurde dann mittels der Kohorten Selektion auf eine Anzahl von 13.000 Akten eingegrenzt.

Für Details zum Starten einer eigenen Beispiel-Anwendung zur Nachvollziehbarkeit der Resultate, siehe Kapitel 5 - Installation):

[6] https://synthetichealth.github.io/synthea/
[7] https://github.com/synthetichealth/synthea/releases/tag/master-branch-latest

Die eigentliche Anfrage wurde ausgeführt via:

```
curl -d @cohort/sample_query.json  -H "Content-Type: application/json"
    http://localhost:8081/query
```

Die referenzierte Beispiel Anfrage lautete dabei:

```
Filter:
- Alle weiblichen Patienten zwischen 30 und 50 Jahren alt.

Werte:
- ID: eindeutige Kennziffer der Akte.
- Given Name: Vorname der Person.
- Family Name: Nachname der Person
- Birth Date: Geburtsdatum
- Gender: (biologisches) Geschlecht.
- Older than 35: Ist die Person älter als 35 Jahre alt.
```

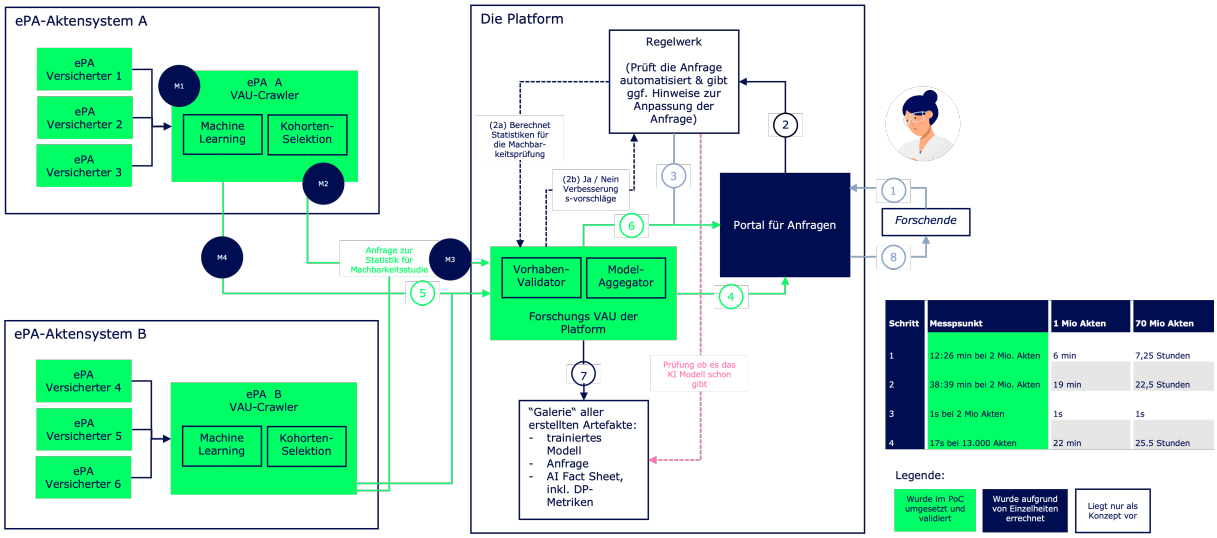
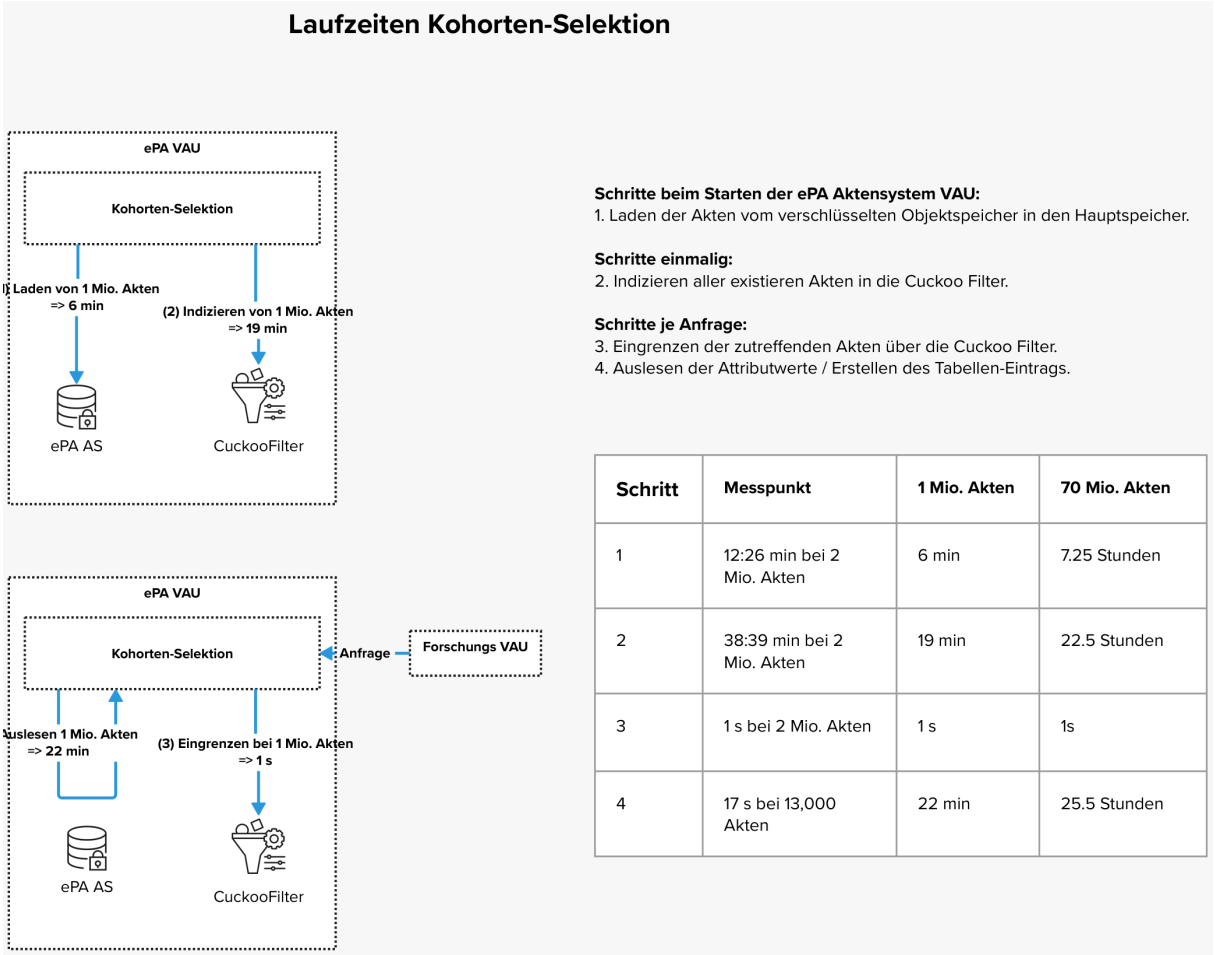


Abbildung 9: Validierung der Skalierbarkeit

Abbildung 9 visualisiert die Evaluierung der Skalierbarkeit. Vier Messpunkte (M1 bis M4) wurden eruiert, welche mehrfach vertestet und deren Ergebnisse gemittelt wurden. Die konkreten Ergebnisse (linke Tabellenspalte) wurden zunächst auf 1. Mio. Datenpunkte normalisiert (mittlere Spalte), und im weiteren Verlauf auf 70. Mio Akten hochgerechnet (rechte Spalte).



Schritt	Messpunkt	1 Mio. Akten	70 Mio. Akten
1	12:26 min bei 2 Mio. Akten	6 min	7.25 Stunden
2	38:39 min bei 2 Mio. Akten	19 min	22.5 Stunden
3	1 s bei 2 Mio. Akten	1 s	1s
4	17 s bei 13,000 Akten	22 min	25.5 Stunden

Abbildung 10: Übersicht der Laufzeiten der Simulation von Kohorten-Selektionen

Wie in Abbildung 10 dargestellt, wird die Berechnung der Laufzeiten extrapoliert aufgrund der vorhandenen Datenpunkte.

4.5 Modelltraining

Zur Messung der Skalierbarkeit des Föderierten Lernens wurde ein Modelltraining mit 73,260,000 Datenpunkte durchgeführt. Dafür wurden die bestehenden Datenpunkte des Beispiel Datensatzes `Stroke Prediction Dataset`[8] der Data Science Plattform “Kaggle” verwendet. Dieser Datensatz enthält 5110 Datenpunkte mit 12 Attributen. Zur Erreichung der gewünschten Datenmenge wurde der Datensatz vervielfacht.

Die Ergebnisse des Modelltraining sowie die konkreten Modell Performanz Werte werden separat im Whitepaper sowie der Visualisierung der trainingsergebnisse bereit gestellt.

[8] https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset/

5 Installation

In diesem Abschnitt werden die nötigen Schritte beschrieben, um eine eigene Installation des Demonstrators bereit zu stellen und die Ergebnisse des Simulationsbericht nachzuvollziehen. Ziel ist die Vorbereitung des PoCs als Demo-Showcase für Präsentationszwecke. Für eine Übersicht der einzelnen Komponenten, ihrer Funktionen und deren Zusammenspiel siehe Architekturübersicht.

Das beiliegende Code Repository enthält den kompletten Proof of Concept, um die Machbarkeit von Federated Learning in Trusted Execution Environments (TEEs) zu demonstrieren. Dieser besteht aus einem FL Server der Client-Verbindungen akzeptiert, um Modell-Trainings-Updates zu aggregieren, sowie zwei Beispiel ePA Aktensystemen. Dieser Funktionale PoC dient der Umsetzung des Architekturkonzepts, um die Qualität und Reife des Konzepts hinsichtlich Sicherheit, Skalierbarkeit und Performance zu demonstrieren und zu zeigen, dass dieser Ansatz auch in einer zukünftigen TI eingesetzt werden kann.

5.1 Lokale Installation via Docker Compose

Die einfachste Variante zum lokalen Starten der Anwendungen geschieht via Docker Compose: zur Orchestrierung der einzelnen Container kann das beiliegende `docker-compose.yml` File benutzt werden. Voraussetzung sind eine aktuelle Version der Container-Plattform `Docker` sowie des mitgelieferten `docker-compose`. Der folgende Befehl startet alle Anwendungen und ihre Abhängigkeiten lokal:

```
1 docker-compose up
```

Beim ersten Ausführen werden die Abhängigkeiten geladen und die Anwendung gebaut, was einige Minuten in Anspruch nehmen kann. Da diese Schritte nur einmalig ausgeführt werden müssen, werden die darauf folgenden Starts deutlich schneller vonstatten gehen. Nach einigen Minuten ist die Anwendung bereit, Anfragen entgegen zu nehmen.

Die Kohorten-Selektion und das Machine Learning benötigen eine Python / Tensorflow Umgebung, sowie einige Python Bibliotheken, welche in der Datei `cohort/requirements.txt` aufgelistet sind. Beim Starten der Applikation via Docker Compose werden diese jedoch über die Container bereit gestellt: eine explizite Installation der Python Bibliotheken ist daher nicht notwendig.

5.2 Übersicht der API Endpunkte

Durch Aufruf der URL `http://localhost:8083/docs` kann die Swagger UI aufgerufen werden, eine grafische Ansicht der verfügbaren API Endpunkte, deren benötigte Parameter sowie Authentifizierung.

5.3 Lokales Testen der Kohorten-Selektion

Nach dem Start der Docker Container müssen zunächst einmalig die Beispieldaten geladen sowie die Indices erstellt werden. Dies kann mittels `curl` wie folgt passieren:

```
curl --request GET \
  --url http://localhost:8083/load
```

Daraufhin wird ein Beispiel Index angelegt mittels:

```
curl --request POST \
  --url http://localhost:8083/index \
  --data '{"filters": ["gender='\''female'\''", "birthDate >= @1968", "
     birthDate <= @1988"],"record_ids": []}'
```

Nun kann die eigentliche Beispiel-Anfrage ausgeführt werden:

```
curl --request POST \
  --url http://localhost:8083/query \
  --header 'content-type: application/json' \
  --data '{"name": "Retrieve all female patients that were born between
      1968 and 1988. Show me the following attributes: Patient ID,
     Given Name, Family Name, Birth Date, Gender. Also include whether
     there they are older than 35.","filters": ["gender='\''female'\''
     ", "birthDate >= @1968", "birthDate <= @1988"],"columns": {"
     Patient ID": "id","Given Name": "name.first().given.first()","
     Family Name": "name.first().family","Birth Date": "birthDate","
     Gender": "gender","Older than 35": "birthDate < @1988"}}'
```

Dieses Beispiel durchläuft eine komplette Anfrage zur Vorbereitung eines Forschungsvorhaben anhand der Beispieldaten und liefert folgendes Ergebnis zurück:

```
Für ihre Kohorten Selektion: ["gender='\''female'\''", "birthDate >=
   @1968", "birthDate <= @1988"] kommen insgesamt 16 Patienten infrage.
```

5.4 Lokales Testen des Föderierten Lernens

Das Training kann gestartet werden über:

```
curl -X GET 0.0.0.0:8081/train
curl -X GET 0.0.0.0:8082/train
```

In den Serverprotokollen sollten nun Ausgaben wie folgende zu sehen sein:

```
docker logs --follow federated-learning-server

2023-11-14 21:28:30.364101: I tensorflow/core/platform/
    cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use
     available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations,
    rebuild TensorFlow with the appropriate compiler flags.
INFO flwr 2023-11-14 21:28:32,916 | app.py:162 | Starting Flower server
    , config: ServerConfig(num_rounds=5, round_timeout=None)
INFO flwr 2023-11-14 21:28:32,936 | app.py:175 | Flower ECE: gRPC
    server running (5 rounds), SSL is disabled
INFO flwr 2023-11-14 21:28:32,937 | server.py:89 | Initializing global
    parameters
INFO flwr 2023-11-14 21:28:32,937 | server.py:276 | Requesting initial
    parameters from one random client
INFO flwr 2023-11-14 21:28:52,810 | server.py:280 | Received initial
    parameters from one random client
INFO flwr 2023-11-14 21:28:52,810 | server.py:91 | Evaluating initial
    parameters
INFO flwr 2023-11-14 21:28:52,810 | server.py:104 | FL starting
DEBUG flwr 2023-11-14 21:28:55,257 | server.py:222 | fit_round 1:
    strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-11-14 21:29:06,555 | server.py:236 | fit_round 1
    received 2 results and 0 failures
WARNING flwr 2023-11-14 21:29:06,559 | fedavg.py:242 | No
    fit_metrics_aggregation_fn provided
DEBUG flwr 2023-11-14 21:29:06,559 | server.py:173 | evaluate_round 1:
    strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-11-14 21:29:07,115 | server.py:187 | evaluate_round 1
    received 2 results and 0 failures
WARNING flwr 2023-11-14 21:29:07,115 | fedavg.py:273 | No
    evaluate_metrics_aggregation_fn provided
DEBUG flwr 2023-11-14 21:29:07,115 | server.py:222 | fit_round 2:
    strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-11-14 21:29:17,253 | server.py:236 | fit_round 2
    received 2 results and 0 failures
DEBUG flwr 2023-11-14 21:29:17,255 | server.py:173 | evaluate_round 2:
    strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-11-14 21:29:17,369 | server.py:187 | evaluate_round 2
    received 2 results and 0 failures
DEBUG flwr 2023-11-14 21:29:17,369 | server.py:222 | fit_round 3:
    strategy sampled 2 clients (out of 2)
DEBUG flwr 2023-11-14 21:29:27,639 | server.py:236 | fit_round 3
    received 2 results and 0 failures
DEBUG flwr 2023-11-14 21:29:27,643 | server.py:173 | evaluate_round 3:
```

```
      strategy sampled 2 clients (out of 2)
25 DEBUG flwr 2023-11-14 21:29:27,774 | server.py:187 | evaluate_round 3
      received 2 results and 0 failures
26 DEBUG flwr 2023-11-14 21:29:27,774 | server.py:222 | fit_round 4:
      strategy sampled 2 clients (out of 2)
27 DEBUG flwr 2023-11-14 21:29:38,122 | server.py:236 | fit_round 4
      received 2 results and 0 failures
28 DEBUG flwr 2023-11-14 21:29:38,126 | server.py:173 | evaluate_round 4:
      strategy sampled 2 clients (out of 2)
29 DEBUG flwr 2023-11-14 21:29:38,285 | server.py:187 | evaluate_round 4
      received 2 results and 0 failures
30 DEBUG flwr 2023-11-14 21:29:38,285 | server.py:222 | fit_round 5:
      strategy sampled 2 clients (out of 2)
31 DEBUG flwr 2023-11-14 21:29:48,599 | server.py:236 | fit_round 5
      received 2 results and 0 failures
32 DEBUG flwr 2023-11-14 21:29:48,603 | server.py:173 | evaluate_round 5:
      strategy sampled 2 clients (out of 2)
33 DEBUG flwr 2023-11-14 21:29:48,712 | server.py:187 | evaluate_round 5
      received 2 results and 0 failures
34 INFO flwr 2023-11-14 21:29:48,712 | server.py:153 | FL finished in
      55.9017290212214
35 INFO flwr 2023-11-14 21:29:48,713 | app.py:225 | app_fit:
      losses_distributed [(1, 0.4915451407432556), (2, 0.2662748396396637)
      , (3, 0.22313693165779114), (4, 0.21079561114311218), (5,
      0.20269928872585297)]
36 INFO flwr 2023-11-14 21:29:48,713 | app.py:226 | app_fit:
      metrics_distributed_fit {}
37 INFO flwr 2023-11-14 21:29:48,713 | app.py:227 | app_fit:
      metrics_distributed {}
38 INFO flwr 2023-11-14 21:29:48,713 | app.py:228 | app_fit:
      losses_centralized []
39 INFO flwr 2023-11-14 21:29:48,713 | app.py:229 | app_fit:
      metrics_centralized {}
40 Stream closed EOF for federated-learning/federated-learning-server-58
      b86c4964-wr52z (federated-learning-server )
```

5.5 Manueller Docker Build

Wer nicht Docker Compose verwenden will oder kann, oder die Applikation bauen, aber nicht starten will, kann als Alternative direkt den Docker Multi-Stage-Build verwenden, welcher sich auf eine gemeinsames Dockerfile stützt, aber das Server- und Client-Image in verschiedenen Phasen des Builds unterteilt. Die einzelnen Container können gebaut werden mit:

```
1 docker build --platform linux/amd64 --target server -t federated-
      learning-server .
2 docker build --platform linux/amd64 --target client -t federated-
      learning-client .
```

(die `--Plattform linux/amd64` stellt x86 Maschinencode her, auch wenn auf einem ARM / Apple M1 gebaut wird).

5.6 Health Platform Test Cluster Run

Um die Images in die IBM Cloud Registry (ICR) zu überführen, markieren Sie sie gemäß dem Health Platform-Namensschema und transferieren sie dann in die ICR:

```
# Tag the images with <region>.icr.io/<my_namespace>/<image_repo>:<tag>
docker tag federated-learning-server:latest de.icr.io/ega_dev/federated-learning-server:v-0.1.0
docker tag federated-learning-client:latest de.icr.io/ega_dev/federated-learning-client:v-0.1.0
# Push to IBM Cloud Registry
docker push de.icr.io/ega_dev/federated-learning-server:v-0.1.0
docker push de.icr.io/ega_dev/federated-learning-client:v-0.1.0
# Verify the upload
ibmcloud cr images | grep federated
```

Sie können nun die Helm Charts bereitstellen:

```
# (Once) Create the namespace
kubectl create namespace federated-learning
# (Once) Copy the ICR Pull Access Token
kubectl get secret icr-image-puller-docker-registry --namespace=vauservices -o yaml | sed 's/namespace: vauservices/namespace: federated-learning/g' | kubectl create -f -
# Install the Helm chart
helm install federated-learning helm/
# Verify the success by checking the status and logs
helm list
helm status federated-learning
kubectl logs -n
```

Sie sollten eine ähnliche Ausgabe sehen wie die im oberen Abschnitt.