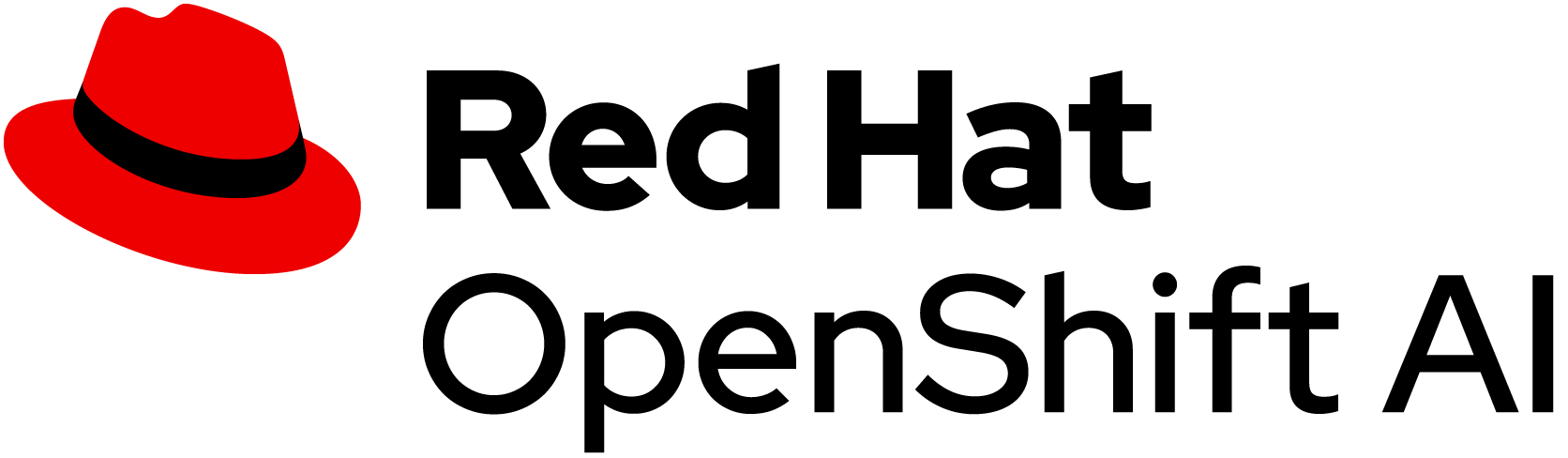
Documentatie On-premise RAG demo



Stagiair: Coen de Vries

Opleiding: Informatica

E-mailadres: [coen.de.vries@HCS-Company.com](mailto:coen.de.vries@hcs-company.com) / [668358@student.inholland.nl](mailto:668358@student.inholland.nl)

Opdrachtgever: HCS Company

K.P. Majoor

Plaats: Amsterdam

Datum: 31-01-2025

Inhoudsopgave

[Begrippenlijst 3](#_Toc188271561)

[1. Chatbot met Retrieval Augmented Generation 4](#_Toc188271562)

[1.1 Large Language Model (LLM) 4](#_Toc188271563)

[1.2 Retrieval Augmented Generation (RAG) 6](#_Toc188271564)

[1.3 LangChain 7](#_Toc188271565)

[1.4 Vector database 9](#_Toc188271566)

[2. On-premise RAG implementatie 10](#_Toc188271567)

[2.1 Logging / Observability 11](#_Toc188271568)

[2.2 Veilig AI in on-promise server 13](#_Toc188271569)

[2.3 HCS server resources 13](#_Toc188271570)

[3. Conclusie 14](#_Toc188271571)

[4. Advies 15](#_Toc188271572)

[5. Tech Stack 16](#_Toc188271573)

# Begrippenlijst

**Artificial Intelligence (AI):** Het vakgebied binnen de informatica dat zich richt op het creëren van machines die taken kunnen uitvoeren die normaal menselijke intelligentie vereisen, zoals redeneren, leren en probleemoplossing.

**Huggingface**: Een open-source platform en bibliotheek voor machine learning en natuurlijke taalverwerking (NLP), dat populaire modellen zoals transformers biedt voor toepassingen zoals tekstclassificatie, vertaling en chatbotontwikkeling.

**IBM Granite**: Een reeks modellen ontwikkeld door IBM, bedoeld voor AI-toepassingen zoals tekstverwerking en data-analyse, en geoptimaliseerd voor gebruik binnen zakelijke en industriële omgevingen.

**Large Language Model (LLM):** Artificial Intelligence dat gebruikmaakt van diepe neurale netwerken om natuurlijke taal te verwerken, waarbij het grote hoeveelheden tekstdata analyseert en genereert om menselijke taal beter te begrijpen en te produceren. Het model kan complexe taken zoals tekstgeneratie, vertaling en samenvattingen uitvoeren door patronen in taal te leren.

**Milvus**: Een open-source vector database die ontworpen is voor het opslaan, beheren en zoeken van hoge-dimensionale vectoren, zoals embeddings die worden gebruikt in AI- en machine learning-toepassingen.

**On-premise**: Een infrastructuur- of software-implementatiemodel waarbij de hardware, servers en applicaties fysiek op locatie van de organisatie worden beheerd en onderhouden, in tegenstelling tot in de cloud.

**OpenShift**: Een familie van softwareproducten voor containerisatie, ontwikkeld door Red Hat. Het is een hybride Cloud platform als een service, gebouwd rond Linux-containers die worden georkestreerd en beheerd door Kubernetes op een fundament van Red Hat Enterprise Linux.

**OpenShift AI**: Een uitbreiding op OpenShift, die helpt bij het ontwikkelen, implementeren en beheren van AI/ML-workloads. Het biedt geïntegreerde tools voor dataverwerking, modeltraining en modelleren.

**PGvector**: Een PostgreSQL-extensie die het mogelijk maakt om vectorgegevens op te slaan en te doorzoeken binnen een relationele database, wat handig is voor AI-toepassingen die relevantie zoekopdrachten gebruiken.

**Retrieval Augmented Generation (RAG):** Een AI-modelarchitectuur die informatie uit externe bronnen ophaalt en combineert met generatiecapaciteiten om nauwkeurigere en contextueel relevante antwoorden te bieden.

**Tracing**: Een techniek in softwaremonitoring waarmee het gedrag van verzoeken of transacties binnen een gedistribueerd systeem wordt gevolgd, door de flow en afhankelijkheden tussen microservices of componenten in kaart te brengen.

**Vector / Embedding**: Een wiskundige representatie van gegevens (zoals tekst of afbeeldingen) in een hoge-dimensionale ruimte, waardoor AI-systemen patronen en relaties kunnen herkennen.

# Chatbot met Retrieval Augmented Generation

In de wereld van kunstmatige intelligentie (AI) worden chatbots steeds vaker ingezet om klantenservicetaken te ondersteunen en de druk op callcenters te verlichten. Een bekend probleem is echter dat deze chatbots vaak niet beschikken over specifieke bedrijfsdata, omdat dergelijke gegevens zijn opgeslagen in privéomgevingen. Hierdoor zijn de antwoorden van AI-modellen vaak niet relevant of toereikend voor klanten. Dit project heeft als doel te onderzoeken hoe Retrieval Augmented Generation (RAG) kan worden ingezet om dit probleem op te lossen door een chatbot te ontwikkelen die wél gebruikmaakt van bedrijfsspecifieke data.

De focus ligt hierbij op het toepassen van RAG binnen het OpenShift platform, een omgeving die breed wordt ingezet voor het beheren en beschikbaar stellen van applicaties. Het project richt zich op het ontwikkelen van een demonstratiemodel dat kan laten zien of en hoe de RAG-methodiek kan bijdragen aan een betere integratie van relevante bedrijfsdata in AI-gestuurde klantenservice.

#### **Vraag:**

* Hoe kan Retrieval Augmented Generation (RAG) gebruikt worden in het OpenShift platform om te werken met de gevoelige data van een bedrijf?

## Large Language Model (LLM)

Large Language Models (LLM’s) zijn AI-modellen die getraind zijn op enorme hoeveelheden tekst en in staat zijn om menselijke taal te begrijpen, te genereren en te verwerken. Deze modellen maken gebruik van neurale netwerken en worden ingezet in verschillende toepassingen zoals tekstgeneratie, vertalingen en chatbots.

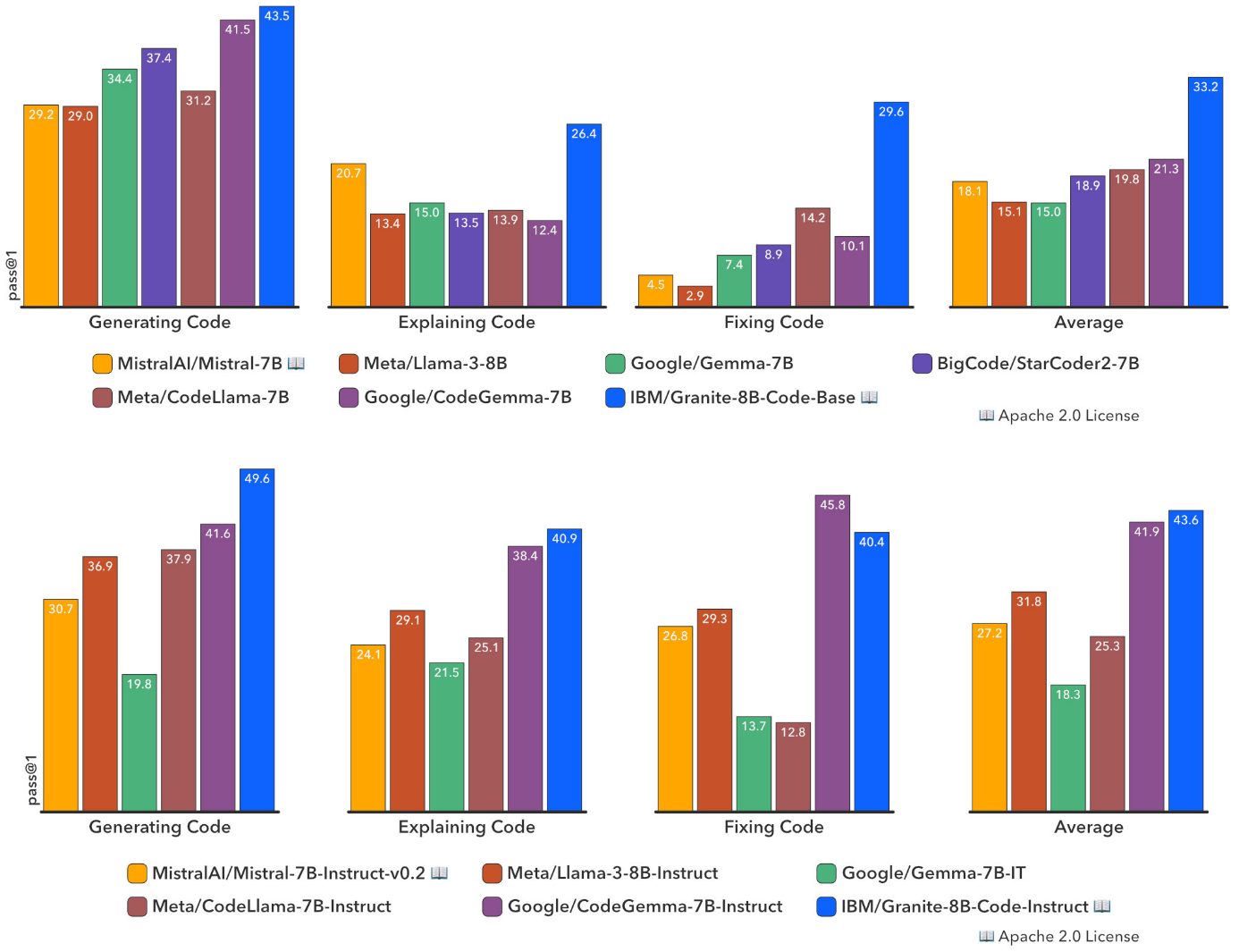
Temperature is een parameter die bepaalt hoe creatief of willekeurig een LLM reageert bij tekstgeneratie. Een lage waarde maakt de uitkomst voorspelbaarder en meer "logisch", terwijl een hogere waarde zorgt voor meer variatie en verrassende antwoorden.

Een token is een klein stuk tekst die door het model wordt verwerkt. Dit kan variëren van een of enkele karakters tot woorden en zinsdelen, afhankelijk van hoe het model is getraind. LLM’s splitsen tekst op in tokens en verwerken deze stapsgewijs. Op basis van deze tokens kan een LLM voorspellen wat het volgende woord moet zijn en taal “begrijpen”.

Quantization is een techniek die de prestaties van AI-modellen verbetert door de precisie van getallen, zoals gewichten, te verlagen van 32-bits naar 16-bits of 8-bits. Deze vermindering van getal precisie leidt tot een efficiëntere uitvoering op hardware met beperkte rekenkracht. De voordelen van quantization omvatten snellere uitvoering en minder geheugengebruik, vaak met een minimaal verlies aan nauwkeurigheid.

Tekstgeneratie verwijst naar de mogelijkheid van een model om nieuwe tekst te creëren op basis van input, context, en eerdere training. LLM’s maken gebruik van voorspellingen modellen om zinnen, paragrafen of zelfs volledige documenten te genereren die kloppend en relevant zijn binnen een gegeven context.

IBM/granite: het granite model is één van de beste opensource modellen beschikbaar. IBM werkt veel samen met andere opensource bedrijven zoals Red Hat en traint ook samen met de opensource gemeenschap het granite model. Het gevolg hiervan is een relatief klein en efficiënt model dat goed is in het verwerken en genereren van tekst.



#### Het figuur laat de prestaties van verschillende taalmodellen zien, met een duidelijke vergelijking tussen de “base” en “instruct” versies van deze modellen. De instruct-modellen zijn specifiek getraind of geoptimaliseerd om beter te presteren op taken die vragen om meer afstemming op menselijke instructies en bruikbare output, terwijl base-modellen algemeen getraind zijn zonder deze specifieke focus.

<https://huggingface.co/ibm-granite>

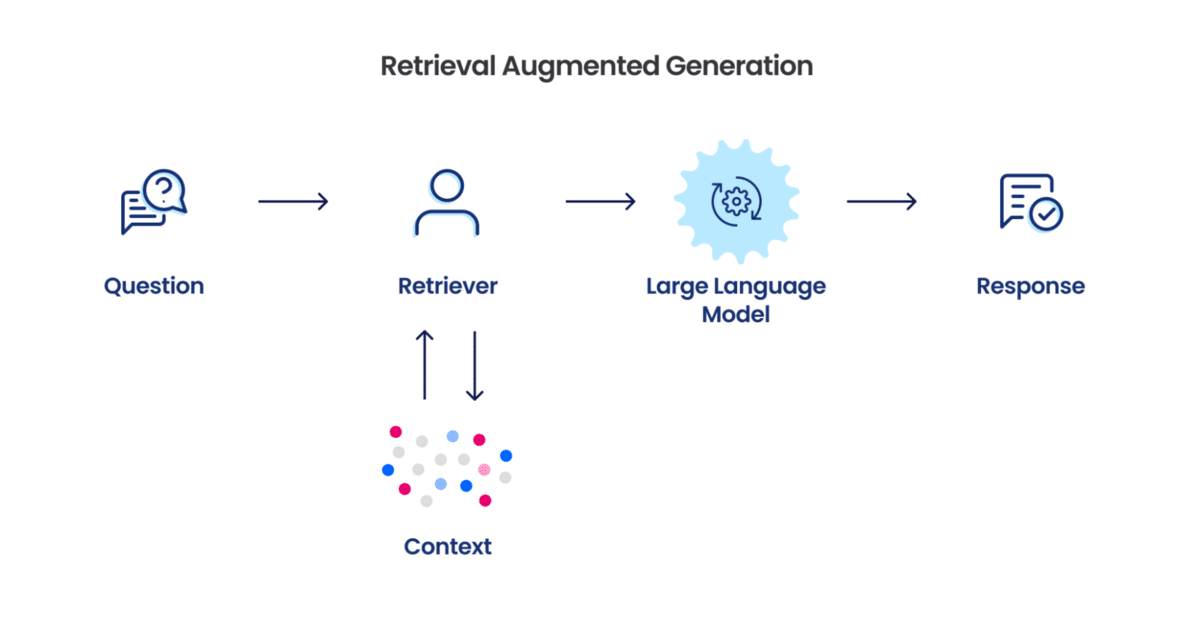
## Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation is een techniek waarbij AI-modellen gebruikmaken van externe kennisbronnen om betere en meer accurate antwoorden te genereren. In plaats van te vertrouwen op de informatie in de getrainde dataset, wordt bij RAG relevante informatie opgehaald uit een database om de output van het model te verbeteren. RAG bestaat uit drie stappen:

Retrieve: Bij retrieval wordt relevante informatie opgezocht uit een database of kennisbron op basis van de input van de gebruiker. Dit helpt het model om betere en meer gefundeerde antwoorden te geven. Deze data in een vector database, hier wordt informatie opgeslagen als een vector. Deze vector kan vervolgens gebruikt worden om een relevantie zoekopdracht te doen. Het wordt vaak gebruikt in RAG-systemen om snel relevante informatie te vinden op basis van de vectoren van het AI-model.

Augment: In de context van RAG betekent augment dat de opgehaalde informatie wordt geïntegreerd met de input prompt van de gebruiker voordat het model een antwoord genereert. Dit verhoogt de accuraatheid en relevantie van de gegenereerde output.

Generate: Nadat relevante informatie is opgehaald en de prompt aangepast is, wordt een nieuwe tekst gegenereerd door het model. Dit proces combineert de kracht van retrieval en de creatieve mogelijkheden van taalmodellen voor een meer contextuele tekstgeneratie.



<https://research.ibm.com/blog/retrieval-augmented-generation-RAG>

## LangChain

LangChain is een framework dat het gebruik van LLM’s vereenvoudigt en uitbreidt door ze te combineren met verschillende tools en workflows. Dit maakt het mogelijk om complexe interacties te creëren met AI-modellen, waaronder documentverwerking, API-integraties en multi-step prompts.

LCEL: staat voor LangChain Expression Language en is een krachtige tool binnen het LangChain-framework. Het stelt gebruikers in staat om complexe bewerkingen en interacties met taalmodellen op een eenvoudige manier te definiëren. LCEL biedt mogelijkheden om logica, functies, en controleflows te integreren in prompts, waardoor de flexibiliteit en controle over de output van LLM’s wordt vergroot.

LlamaCppPython: De kracht van LlamaCppPython ligt in de combinatie van de snelheid van C++ met de eenvoud en flexibiliteit van Python. In LangChain kunnen functies zoals prompt templates, LCEL en chat buffer memory gemakkelijk worden geïntegreerd met LLaMA-modellen via LlamaCppPython. Dit zorgt voor efficiënte en krachtige AI-oplossingen, terwijl ontwikkelaars profiteren van de brede mogelijkheden van het Python-ecosysteem. De C++ gereedschappen zijn al gecompileerd; er hoeft alleen maar via python code gebruik van gemaakt te worden.

Prompt templates: Prompt templates in LangChain zijn sjablonen die gebruikt worden om de input voor een LLM te structureren. Deze sjablonen helpen om consistente en effectieve interacties met het model te waarborgen door standaard prompts of vragen te definiëren. Deze sjablonen maken onderscheid tussen het systeem en de gebruiker om het AI-model een beter idee te geven over de context en de uit te voeren instructies.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Context voor het AI model

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Instellingen voor het AI model

Chat buffer memory is een functionaliteit binnen LangChain die ervoor zorgt dat het model toegang heeft tot eerdere interacties binnen een sessie. Dit zorgt ervoor dat het model context onthoudt en consistent blijft in lange gesprekken, wat vooral nuttig is in chatbotapplicaties. De toegevoegde waarde is dat op deze manier het AI model de eerdere vragen en antwoorden kan raadplegen om betere antwoorden te geven, ook helpt het om het model context te geven wanneer de gebruiker het over ‘dit’ heeft.

Response streaming is een manier van asynchroon programmeren om niet te wachten tot een heel resultaat behaald is, zoals het genereren van een antwoord. Het AI-model stuurt dus woord voor woord, of token voor token, een stukje naar de frontend. Dit is een goede techniek om feedback aan een eindgebruiker te geven. Zeker wanneer het genereren van een antwoord langer duurt. Omdat het genereren van antwoorden op een laptop lang duurt kan je te maken krijgen met timeout errors. Standaard gebeurt dit met vijf minuten. Met response streaming blijft de verbinding wel openstaan aangezien er iedere paar seconden een nieuw stukje van het antwoord wordt ontvangen.

Omdat response streaming asynchroon antwoorden verwerkt wordt informatie stukken sneller verwerkt. Dit komt doordat er geen tijd besteed wordt aan het bufferen van informatie en er geen latency ontstaat bij het versturen van grotere antwoorden. Bij het IBM granite model gebruikt voor de demo, scheelt dit ongeveer 40 á 50 procent.

<https://www.langchain.com>

## Vector database

Om relevante documenten op te kunnen zoeken moeten deze omgezet zijn naar vectoren. Als eerste optie is er met de Milvus vectordatabase gewerkt. Dit omdat het een populaire optie is en veel demonstraties hier mee werken maar ook omdat het vooronderzoek van Bram Terlouw hier mee werkte. Bij dit onderzoek is de database uitgebreid om niet in het geheugen te werken, maar in een pod in het OpenShift cluster.

Het opzetten van de database in een lokale container ging verassend snel. Met behulp van een paar demonstraties en de handige interface van Podman Desktop was de database zo opgezet en waren documenten snel toegevoegd. Er ontstonden pas problemen op het moment dat de database naar OpenShift verplaatst moest worden. Hier kom je verschillen tegen in permissies van containers, ook heeft OpenShift beveiligingsregels toegevoegd om ervoor te zorgen dat databases niet vanaf buitenaf benaderd kunnen worden, hierdoor leek het voor een lange tijd alsof de database niet werkte in het OpenShift cluster. De oplossing hiervoor was het implementeren van een API die de connectie maakt met de database en vanaf buiten het cluster te benaderen is. Door zelf een API te implementeren kan je zelf de veiligheid controleren en filteren welke data je wel en niet wilt doorsturen. Door gebruik te maken van de API kan er lokaal gewerkt worden aan de AI en RAG-implementatie en kan er tegelijkertijd documenten opgehaald worden uit de cloud.

<https://milvus.io/docs>

#### PostgreSQL PGvector extensie

Omdat PostgreSQL opensource is en veel gebruikt wordt door de klanten van HCS is er naast Milvus ook onderzoek gedaan naar de PGvector extensie. Deze extensie zorgt ervoor dat er naast de standaard datatypes zoals integer of varchar ook een vector-datatype beschikbaar wordt gesteld. Nadat er programmatisch een vector/embedding is gegenereerd kan met de extensie dit opgeslagen worden.

Wanneer de nieuwe kolom met de vectors is toegevoegd kunnen hier ook nieuwe vergelijkingen mee uitgevoerd worden. Zo kan het ‘similarity searches’ uitvoeren op basis van Cosinusafstand (cosine similarity), Euclidische afstand (euclidean distance) en Innerlijke product (inner product). Dit zijn manieren om met vectors natuurlijke tekst op relevantie te vergelijken.

Een voordeel van PGvector is dat het naadloos aansluit op een al bestaande database. De extensie hoeft alleen toegevoegd te worden en geactiveerd te worden. Hierna kan er meteen een nieuw vector-kolom toegevoegd worden aan het tabel.

De extensie is beschikbaar als containerimage en kan handmatig toegevoegd worden aan een zelfgemaakte PostgreSQL image. Let op dat de extensie eerst geactiveerd moet worden voordat het gebruikt kan worden:

*CREATE EXTENSION vector;*

<https://github.com/pgvector/pgvector>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Aspect | Milvus | | PostgreSQL met pgvector | | --- |  |  | | --- | |  | |
| Specialisatie | Gespecialiseerde vectordatabase | Relationele database met vector-extensie |
| Functionaliteit | Geavanceerde vectorzoekalgoritmen (HNSW, IVF) | Basis vectorzoek- en vergelijkingsmogelijkheden |
| Schaalbaarheid | Hoog, ontworpen voor grote vectordatasets | Beperkt bij grootschalige vectorbewerkingen |
| Performance | Lage latentie, geoptimaliseerd voor vectordata | Goed voor kleine/middelgrote vectordatasets |
| Integratie in OpenShift | Vereist aparte setup en resources | Naadloze integratie binnen bestaande PostgreSQL-installatie |
| Gebruiksgemak | Nieuwe tools en leercurve nodig | Eenvoudig als je al PostgreSQL gebruikt |
| Kosten | Extra infrastructuur en onderhoud nodig | Kostenbesparend bij bestaande PostgreSQL |
| Ideaal gebruik | High-performance AI/ML-toepassingen | Aanvulling op bestaande relationele toepassingen |

# On-premise RAG implementatie

Veel bedrijven willen Artificial Intelligence (AI) gebruiken om hun processen te verbeteren, bijvoorbeeld door grote hoeveelheden informatie snel te doorzoeken en samen te vatten. Een bekende aanpak hiervoor is Retrieval Augmented Generation (RAG). Hierbij gebruikt een AI-model een context met eigen data om relevante antwoorden te genereren zonder het model te trainen op deze data, zoals aangetoond in de eerste opdracht van dit document. Maar als bedrijven dit willen doen, zijn er zorgen over de veiligheid van gegevens en hoe alles goed werkt binnen hun bestaande IT-omgeving. De cloud is een optie, maar voor veel bedrijven is gegevensbescherming reden genoeg om dit liever in hun eigen datacenter te doen.

In opdracht 2 gaat er daarom gekeken worden hoe en of er binnen OpenShift een systeem gerealiseerd kan worden om AI te gebruiken om de bedrijfsprocessen te verbeteren zonder risico te lopen dat de data publiekelijk toegankelijk wordt.

De opdracht is om een werkende demonstratie te ontwikkelen die laat zien hoe OpenShift, AI en RAG ondersteunt binnen een eigen datacenter. Dit moet duidelijk maken hoe veiligheid wordt gegarandeerd, hoe gegevens worden verwerkt en waar deze gegevens vandaan komen en heen gaan.

#### **Vraag:**

* Is RAG in een on-premise omgeving toe te passen en is dit een veilige manier om met gevoelige gegevens en AI om te gaan?

## Logging / Observability

Om meer inzicht te krijgen in de stappen die gezet worden in de RAG-applicatie, en aan te tonen dat AI veilig te gebruiken is in combinatie met gevoelige data. Worden alle stappen individueel gelogd en opgeslagen in een database.

Afbeelding met schermopname, zwart, ontwerp

Automatisch gegenereerde beschrijving

Omdat voor deze stageopdracht gelimiteerde middelen beschikbaar zijn kan er niet gebruik gemaakt worden van grote en bekende observability stacks zoals EFK-Stack. Wanneer er in een productieomgeving wel genoeg middelen beschikbaar zijn kan dit goed toegepast worden om de prestaties van het RAG systeem te controleren. Voor de demo is er een logging systeem gebouwd met het idee van tracing, hier is een opensource implementatie van te vinden in Grafana.

Traces zorgen ervoor dat de lifecycle van een verzoek in kaart wordt gebracht. Dit wordt gedaan door het eerste verzoek een uniek id te genereren en in iedere vervolg stap deze id mee te geven, zoals met de implementatie van het correlation\_id in de demo. Het verschil tussen de demo en de Grafana implementatie is dat de unieke id’s beheerd worden vanuit een observability applicatie en niet vanuit de eerste container waar het verzoek ontstaat.

<https://medium.com/@tech_18484/simplifying-kubernetes-logging-with-efk-stack-158da47ce982>

<https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/traces/>

Afbeelding met tekst, schermopname, Lettertype, ontwerp

Automatisch gegenereerde beschrijvingAfbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijvingHet tonen van de logs wordt gedaan via dezelfde website waar de demonstratie uitgevoerd wordt. Hiervoor is een nieuwe pagina toegevoegd die alle logs ophaalt uit de database en dit in een logische structuur presenteert.

## Veilig AI in on-promise server

OpenShift biedt een uitgebreide reeks beveiligingsmechanismen om de veiligheid van applicaties en infrastructuur te garanderen. Dit begint op infrastructuurniveau, waar OpenShift draait op beveiligde omgevingen zoals bare metal, virtuele machines of cloudplatformen. Het ondersteunt veilige protocollen zoals HTTPS en integreert met de beveiligingssystemen van de onderliggende infrastructuur. Daarnaast kunnen SCAP-beveiligingsprofielen (Security Content Automation Protocol) worden gebruikt om naleving van beveiligingsstandaarden zoals CIS en NIST te waarborgen.

Containerbeveiliging: OpenShift biedt krachtige tools om kwetsbaarheden in container-images te detecteren. Met tools zoals Red Hat Quay en OpenShift Advanced Cluster Security (ACS) worden container-images gescand, terwijl image signing en verificatie ervoor zorgen dat alleen vertrouwde images worden gebruikt. Tijdens runtime monitort OpenShift verdachte activiteiten en afwijkingen in het gedrag van containers, zoals onverwachte netwerkverzoeken of ongeautoriseerde wijzigingen in bestanden.

Netwerkbeveiliging: is een kernonderdeel van OpenShift. Door middel van Kubernetes NetworkPolicies kan het verkeer tussen pods worden gereguleerd om ongeautoriseerde communicatie te voorkomen. Verder maakt het gebruik van service mesh-technologie, zoals Istio, waarmee end-to-end encryptie en zero-trust communicatie tussen microservices wordt gerealiseerd. OpenShift zorgt er ook voor dat data tijdens transport standaard wordt versleuteld met TLS.

Role-Based Access Control (RBAC): zorgt ervoor dat toegang tot resources wordt beperkt op basis van rollen, terwijl integratie met systemen zoals LDAP, Active Directory en OAuth een naadloze gebruikersauthenticatie mogelijk maakt. Projecten (namespaces) binnen OpenShift zijn standaard geïsoleerd om de toegang tot resources verder te beperken en risico's te minimaliseren.

CI/CD-pipelines: zijn ontworpen met beveiliging in gedachten. Tools zoals Tekton en Jenkins zijn geïntegreerd met het platform, waardoor beveiligingscontroles zoals imagevalidatie en vulnerability scanning, een vast onderdeel zijn van het build- en deploymentproces. Beleidsregels kunnen worden afgedwongen om ervoor te zorgen dat alleen veilige en conforme workloads worden uitgerold.

Tot slot implementeert OpenShift de principes van zero trust, waarbij minimale privileges standaard worden afgedwongen. Workloads worden geïsoleerd op basis van hun specifieke eisen en gevoeligheid, wat het risico van aanvallen verder vermindert. Door deze uitgebreide aanpak biedt OpenShift een robuust en geïntegreerd beveiligingsmodel dat voldoet aan de eisen van moderne, container gebaseerde applicaties.

## HCS server resources

Het granite model gebruikt voor de demo is 5gb. Dit hele bestand moet in het RAM geheugen geladen worden om het model te laten werken. Voor AI en LLM is dit een heel klein model, er is hiervoor gekozen omdat de demo op een laptop ontwikkeld moet worden en het OpenShift cluster alleen toegang heeft tot CPU’s.

Omdat het model klein is heeft het weinig kennis en is de kwaliteit ook duidelijk minder goed dan populaire AI-modellen. Toch is er duidelijk aan te tonen dat het model werkt met de RAG methodiek, dus is de demo goed uit te voeren.

# Conclusie

#### **Vraag:**

* Hoe kan Retrieval Augmented Generation (RAG) gebruikt worden in het OpenShift platform om te werken met de gevoelige data van een bedrijf?

#### **Antwoord:**

RAG kan effectief worden toegepast binnen het OpenShift-platform door gebruik te maken van een combinatie van containers, een backend API, en een vector database zoals Milvus of PGvector. Het proces van RAG bestaat uit drie belangrijke stappen:

1. Relevante data ophalen: De chatbot zoekt in een vector database naar documenten die relevant zijn voor de vraag van de gebruiker.
2. Context toevoegen: De gevonden data wordt toegevoegd aan de vraag als extra context voor het AI-model.
3. Antwoord genereren: Het AI-model gebruikt de aangeleverde context om een specifiek en relevant antwoord te formuleren.

De infrastructuur van OpenShift helpt om deze processen schaalbaar en efficiënt te maken. Verschillende componenten, zoals de frontend, backend, en databases, worden beheerd in containers, waardoor ze eenvoudig te beheren en te updaten zijn. Het AI-model voor deze opdracht is klein (5gb) zodat het op een laptop en het cluster van HCS kan werken. In een productieomgeving kan er met behulp van krachtigere OpenShift-resources een model van betere kwaliteit geïmplementeerd worden.

Met RAG kunnen chatbots antwoorden genereren die zijn afgestemd op specifieke bedrijfsdata, zonder dat het AI-model zelf volledig opnieuw getraind hoeft te worden. Hierdoor bespaart deze aanpak tijd en middelen, terwijl het ook zorgt voor betere resultaten dan generieke chatbots.

Kortom, RAG biedt een flexibele en effectieve oplossing om AI in te zetten met gevoelige bedrijfsdata, terwijl OpenShift zorgt voor een veilige, schaalbare en goed beheersbare omgeving om deze technologie te hosten. Deze combinatie kan bedrijven helpen om hun klantenservice te verbeteren en processen te optimaliseren.

#### **Vraag:**

* Is RAG in een on-premise omgeving toe te passen en is dit een veilige manier om met gevoelige gegevens en AI om te gaan?

#### **Antwoord:**

RAG kan veilig en goed werken in een on-premise omgeving, zolang de juiste stappen worden gevolgd om data te beschermen. De demonstratie heeft laten zien dat het mogelijk is om een RAG-systeem in een eigen datacenter op te zetten. Hierbij blijft alle gevoelige data binnen de veilige grenzen van het datacenter.

Met OpenShift en een simpel maar doeltreffend loggingsysteem wordt inzichtelijk gemaakt hoe data wordt verwerkt en welke stappen worden doorlopen. Door unieke ID’s te gebruiken, kunnen alle acties goed worden gevolgd. Dit zorgt ervoor dat er transparantie is en dat alles veilig blijft.

Hoewel in deze demonstratie eenvoudiger tools zijn gebruikt dan in grote productiesystemen, zoals de EFK-stack of Grafana, laat het systeem zien dat de belangrijkste doelen worden behaald: veiligheid, controle en bruikbaarheid. De demonstratie heeft ook bewezen dat het hele systeem kan draaien zonder dat data via internet gaat, wat cruciaal is voor gevoelige informatie.

Kortom, het is goed mogelijk om RAG veilig in een eigen datacenter te gebruiken. Het helpt bedrijven om AI in te zetten zonder dat de veiligheid van hun data in gevaar komt. Wel moet er rekening gehouden worden met de hardware eisen van het AI-model en de kosten die daar bij komen kijken.

# Advies

#### PGvector vs Milvus

Voor toepassingen in een OpenShift-cluster waarbij de focus ligt op schaalbare en high-performance vectorzoekopdrachten, zoals AI/ML-modellen of multimediatoepassingen, is Milvus de betere keuze vanwege zijn gespecialiseerde algoritmen en optimalisaties. Als vectorfunctionaliteit slechts een aanvulling is op een bestaande relationele database-infrastructuur, of als de datasets en workload beperkt zijn, biedt PostgreSQL met pgvector een kostenefficiënte en eenvoudig te implementeren oplossing, vooral als PostgreSQL al in gebruik is binnen het cluster. De keuze hangt af van de balans tussen complexiteit, prestatie-eisen en operationele kosten.

#### AI en resource gebruik

Voor een efficiënte inzet van AI in een eigen datacenter is goed resourcebeheer cruciaal, met investering in krachtige CPU's, GPU's, snelle opslag zoals NVMe SSD's, en een robuust netwerk. Containerisering en tools zoals OpenShift bieden flexibiliteit en schaalbaarheid. Beveilig gevoelige data via strikte toegangscontrole, netwerksegmentatie en lokale verwerking.

De keuze tussen on-premise en cloud hangt af van controle, kosten en flexibiliteit: on-premise biedt meer controle en lagere operationele kosten op lange termijn, maar vereist hoge initiële investeringen, terwijl de cloud schaalbaarheid en toegang tot nieuwe technologieën biedt, maar met hogere kosten en mogelijke data-compliance zorgen. On-premise is ideaal voor hoge databeveiliging en voorspelbare werklasten, terwijl de cloud beter is voor flexibele, snel veranderende workloads.

#### Gevaar RAG en onbevoegde toegang tot data

Bij het gebruik van Retrieval Augmented Generation (RAG) met gevoelige data is het grootste risico datalekken en ongeautoriseerde toegang tijdens de verwerking en opslag van data. In de cloud biedt schaalbaarheid, maar de veiligheid is afhankelijk van de provider en vereist strikte compliance met dataprivacyregels. On-premise geeft meer controle over gevoelige data, maar vereist robuuste beveiligingsmaatregelen zoals netwerksegmentatie en versleuteling.

Ongeacht de locatie is het essentieel om toegang goed te beheren, logging toe te passen en AI-processen te controleren. Daarnaast bestaat bij RAG-chatbots het risico dat gebruikers toegang krijgen tot niet-begunstigde informatie, wat kan worden voorkomen door Role-Based Access Control (RBAC), strikte zoekfilters, data-anonimiseren en regelmatige beveiligingstests. Met deze maatregelen kan RAG veilig worden ingezet, zowel on-premise als in de cloud.

# Tech Stack

* **VueJs frontend**: website om RAG demo interactief uit te voeren.
* **Python Backend (Flask)**: Backend om met databases te communiceren.
* **PostgreSQL database**: gebruikersdata zoals account en verzekerde auto(s). En logs van de RAG demo.
* **Milvus vectordatabase**: Om verzekeringsdocumenten in op te slaan met een vector/embedding.
* **PGvector**: extensie voor PostgreSQL om het vector datatype aan een kolom mee te kunnen geven, wordt gebruikt om Milvus te vervangen.
* **IBM/Granite model**: het Large Language Model (LLM) gebruikt om antwoorden mee te genereren. <https://huggingface.co/bartowski/granite-3.0-8b-lab-community-GGUF>
* **Podman**: om alle applicaties te containeriseren en via Red Hat extensies beschikbaar te stellen aan OpenShift.
* **OpenShift**: Dev Sandbox voor het testen van de verschillende onderdelen van de applicatie, het HCS cluster om de complete demo uit te voeren.