Red Hat Summit

Connect

Drei Schritte zur smarten App

LLM-Integration mit Quarkus leicht gemacht

Ruben Verma
Specialist Solution Architect
ruverma@redhat.com

Georg Modzelewski
Specialist Solution Architect
georg@redhat.com

Ruben Verma

Specialist Solution Architect
Red Hat

Georg Modzelewski

Specialist Solution Architect
Red Hat

Agenda

Einführung
Quarkus & LangChain4j

0. LLM Deployment mit Podman AI

1. Quarkus vorbereiten
   Dependencies einbinden

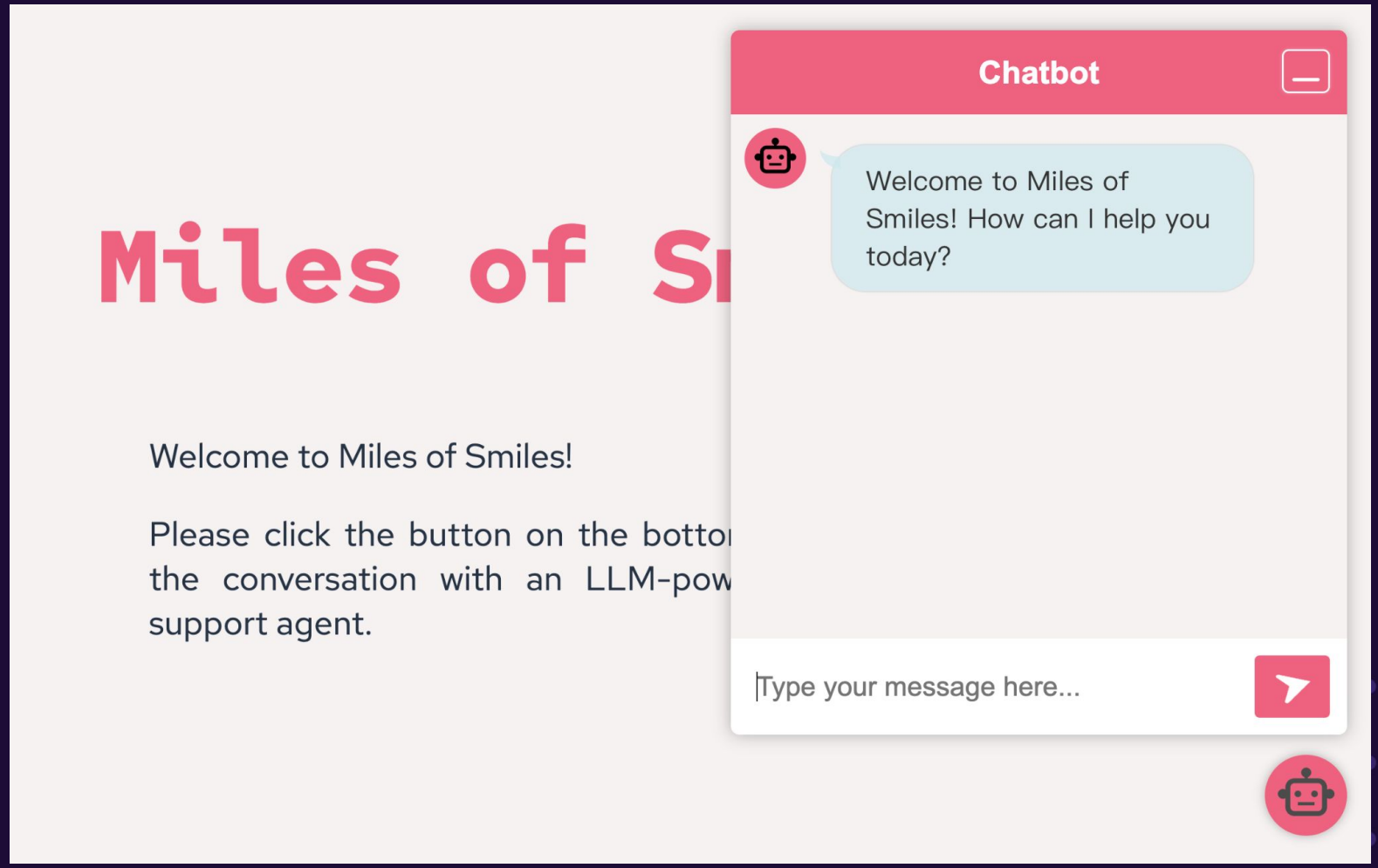
2. Konfiguration
   Finale Anpassungen der Eigenschaften

3. LLM-Service implementieren
   Nutzung von `@RegisterAiService`


AI


cat
puccino
AI
for
DUMMIES

Ziel

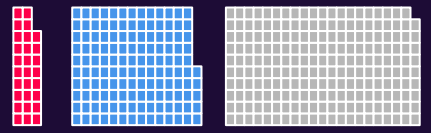

Miles of Sm
Welcome to Miles of Smiles!
Please click the button on the botto
the conversation with an LLM-pow
support agent.
Chatbot
Welcome to Miles of Smiles! How can I help you today?
Type your message here...


AI
Quarkus
Langchain4J


QUARKUS®

Quarkus

TIOBE : #1
IEEE : #1
SlashData : #2
RedMonk : #2



Solid Foundation

Java ist konstant unter den Top 3 der Programmiersprachen mit 7-10 Millionen Entwicklern.

Stunning Performance

Optimiert für nativen Speicher und Startzeit, ermöglicht dies höhere Dichte, Leistung, Elastizität und geringere Kosten.

Toolchain

OpenShift Developer Console, Dev Spaces, Project Generator, Live-Reload für schnellen Inner-Loop-Workflow und Tekton Pipelines Integration.

Community

Umfangreiche Erweiterungen verbinden Ihre Anwendungen mit Top-Technologien wie Camel, Jaeger, Prometheus, Istio, Kafka u.v.m.


langchain4J

LangChain (Python, JS)

LangChain

Vector Stores

Prompts

Document Loaders

Models

Chains

Agents

LangChain4j

LangChain4j

Vector Stores

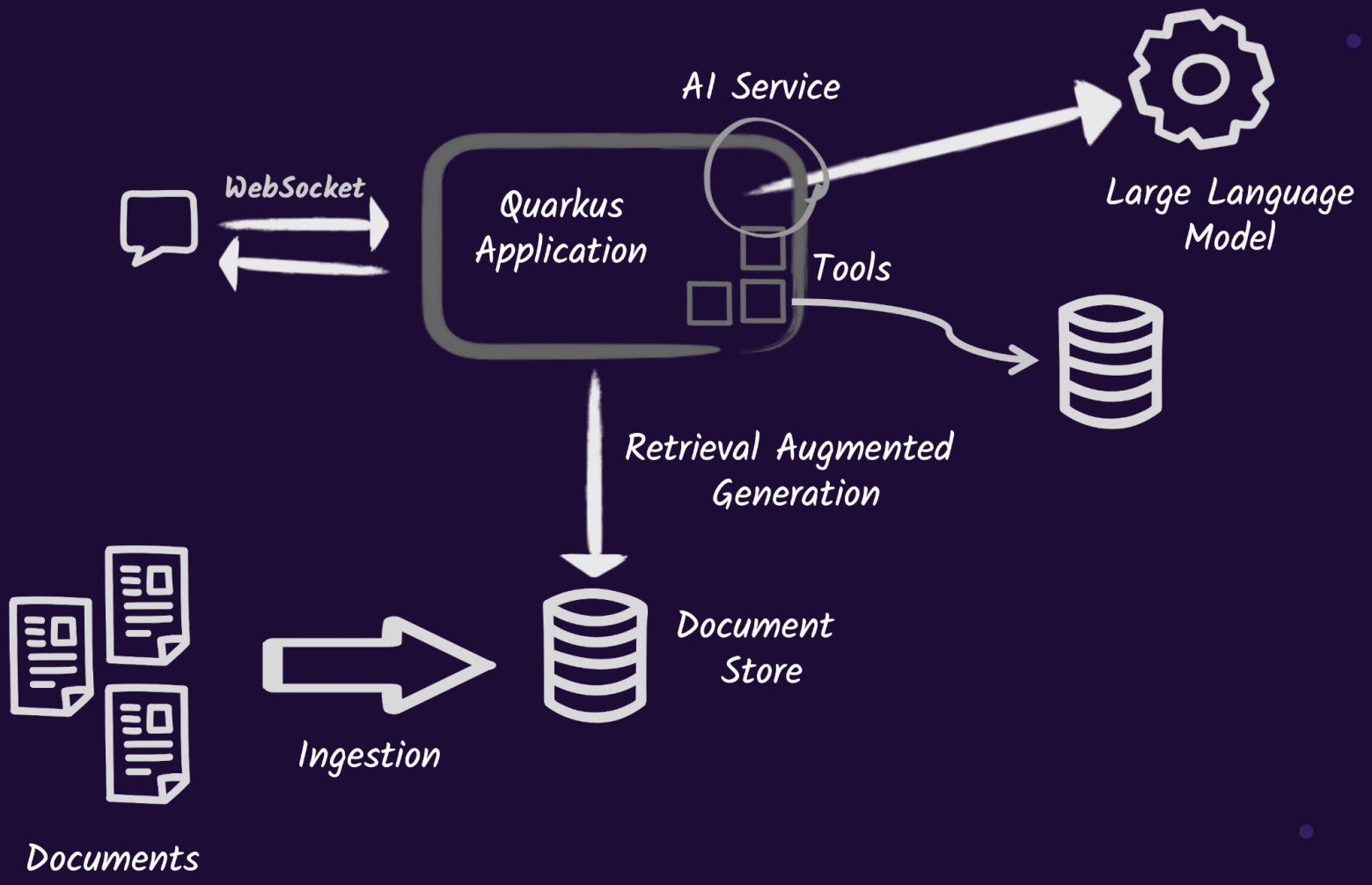
Prompts

Document Loaders

Models

Chains

Agents


AI Service
WebSocket
Quarkus Application
Tools
Large Language Model
Retrieval Augmented Generation
Document Store
Ingestion
Documents

Damals... Pre-LLM

Frontend

```java
package dev.langchain4j.quarkus.workshop;

import dev.langchain4j.service.SystemMessage;
import io.quarkiverse.langchain4j.
RegisterAiService;
import jakarta.enterprise.context.SessionScoped;

@SessionScoped
@RegisterAiService
public interface CustomerSupportAgent {

    @SystemMessage("You are a helpful agent")
    String chat(String userMessage);
}
```

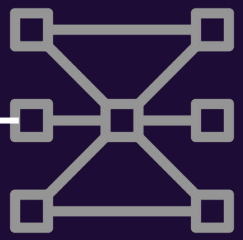
Backend

Heute

Frontend

```java
package dev.langchain4j.quarkus.workshop;

import dev.langchain4j.service.SystemMessage;
import io.quarkiverse.langchain4j.
RegisterAiService;
import jakarta.enterprise.context.SessionScoped;

@SessionScoped
@RegisterAiService
public interface CustomerSupportAgent {

    @SystemMessage("You are a helpful agent")
    String chat(String userMessage);
}
```

Backend



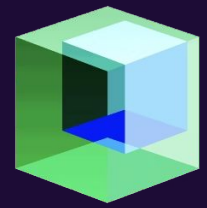
LLM

Quarkus

REST-API

```java
package dev.langchain4j.quarkus.workshop;

import dev.langchain4j.service.SystemMessage;
import io.quarkiverse.langchain4j.
RegisterAiService;
import jakarta.enterprise.context.SessionScoped;

@SessionScoped
@RegisterAiService
public interface CustomerSupportAgent {

    @SystemMessage("You are a helpful agent")
    String chat(String userMessage);
}
```

LangChain4j



Podman Desktop

Frontend

Backend

Schritt 0: LLM Deployment mit Podman AI

Frontend

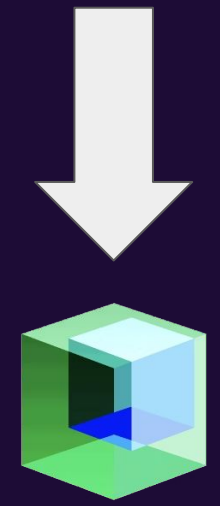
REST-API

Quarkus

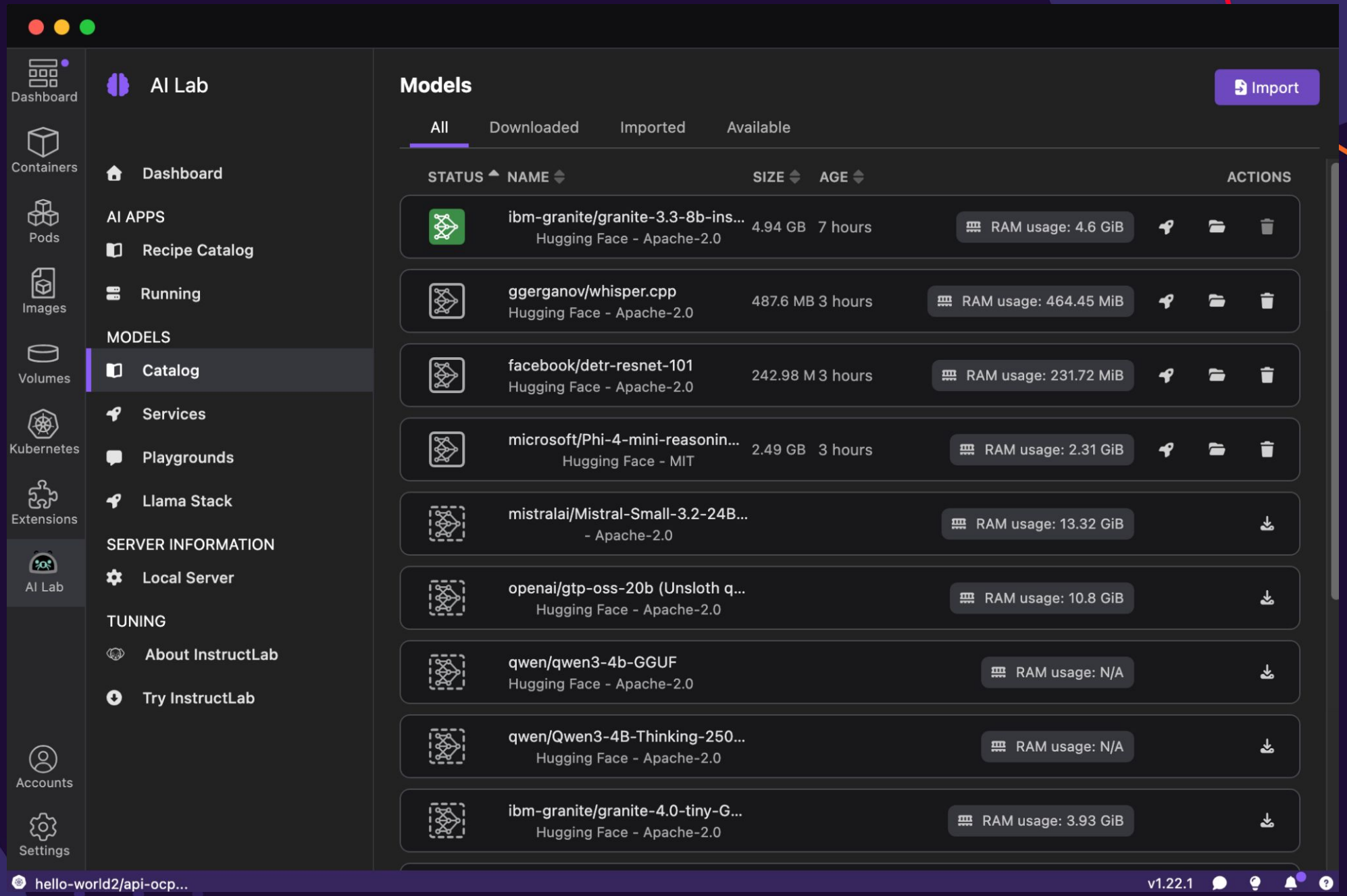
```java
package dev.langchain4j.quarkus.workshop;

import dev.langchain4j.service.SystemMessage;
import io.quarkiverse.langchain4j.
RegisterAiService;
import jakarta.enterprise.context.SessionScoped;

@SessionScoped
@RegisterAiService
public interface CustomerSupportAgent {

    @SystemMessage("You are a helpful agent")
    String chat(String userMessage);
}
```

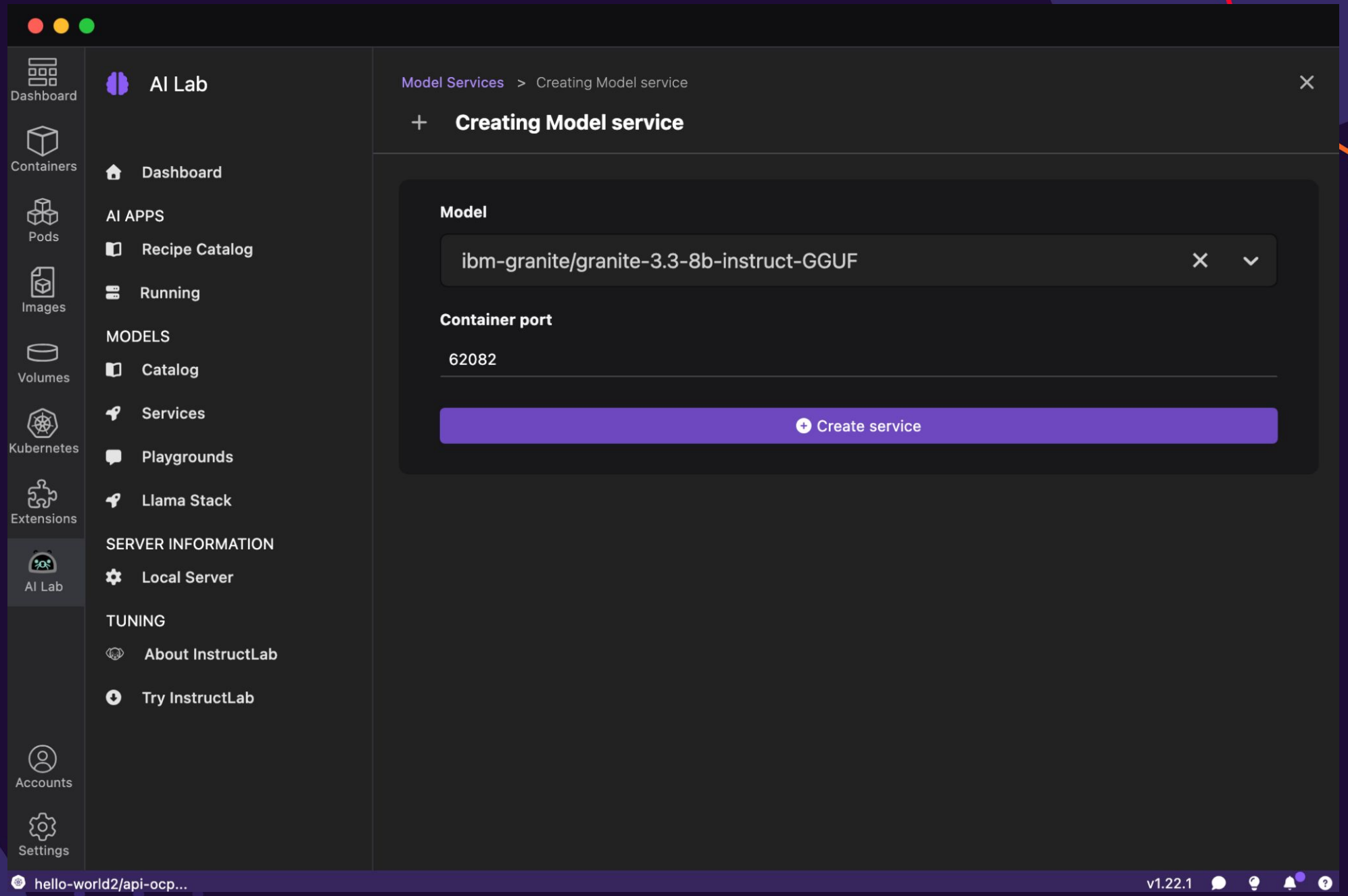
Backend

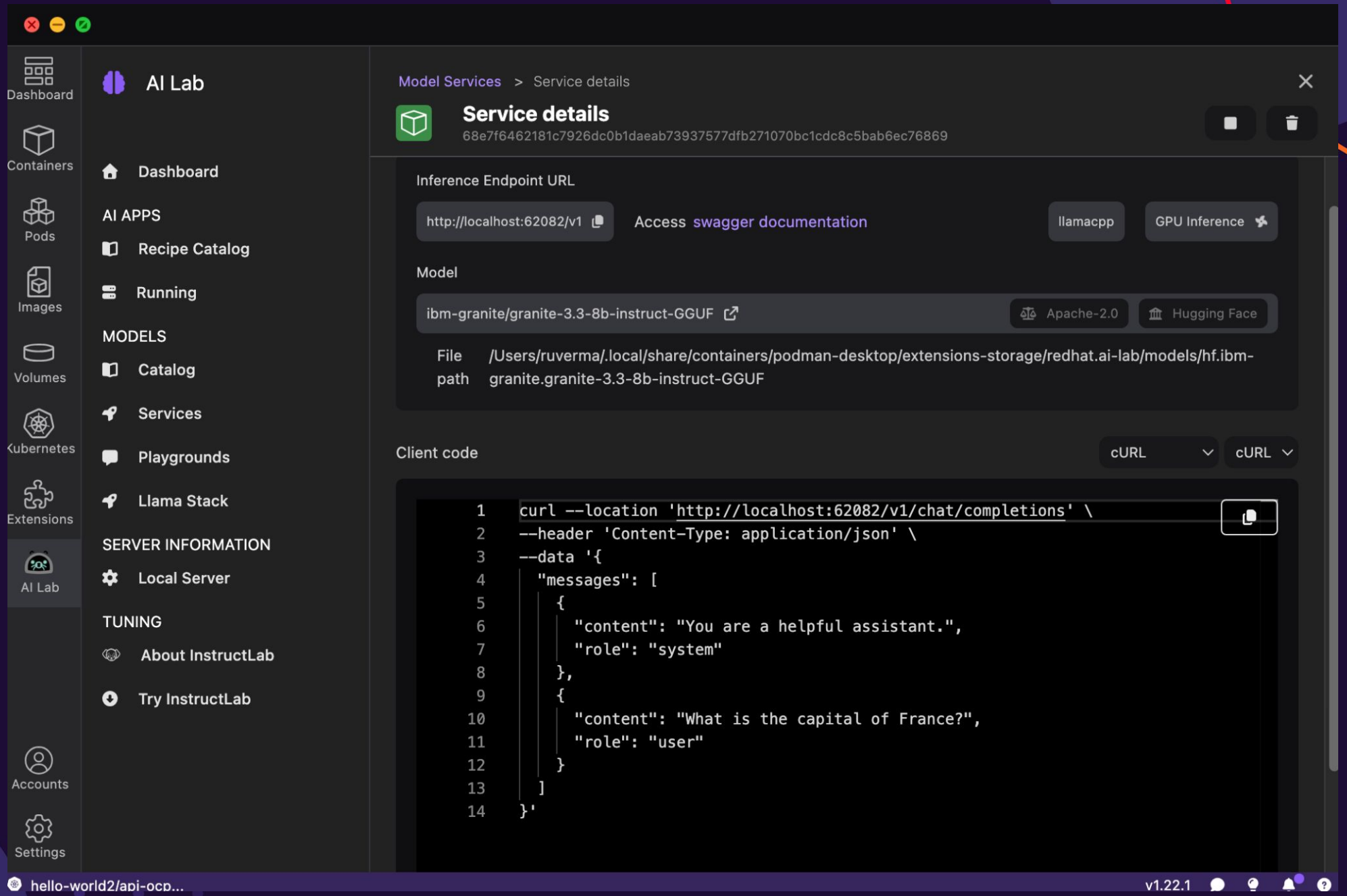
LangChain4j

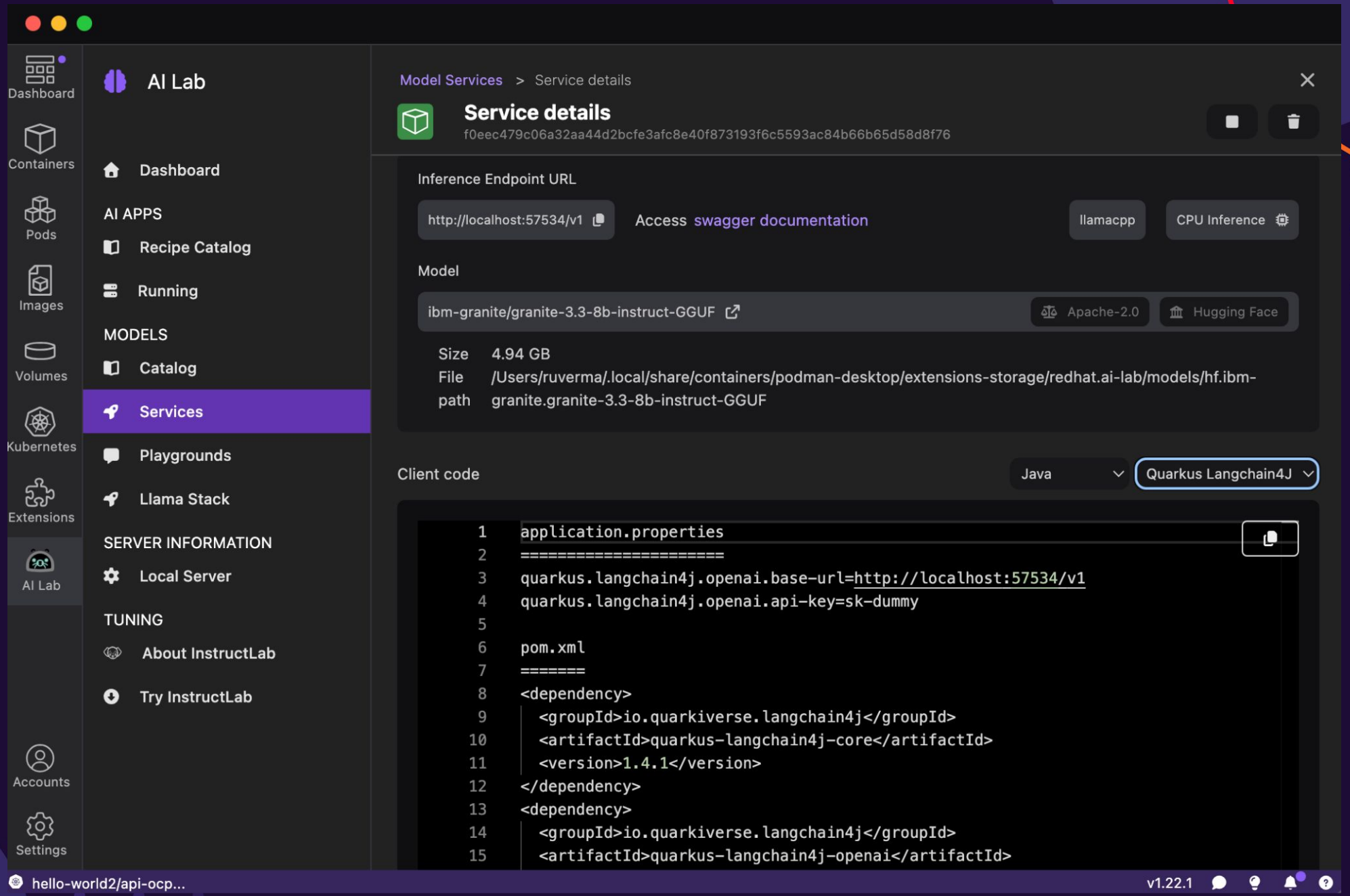


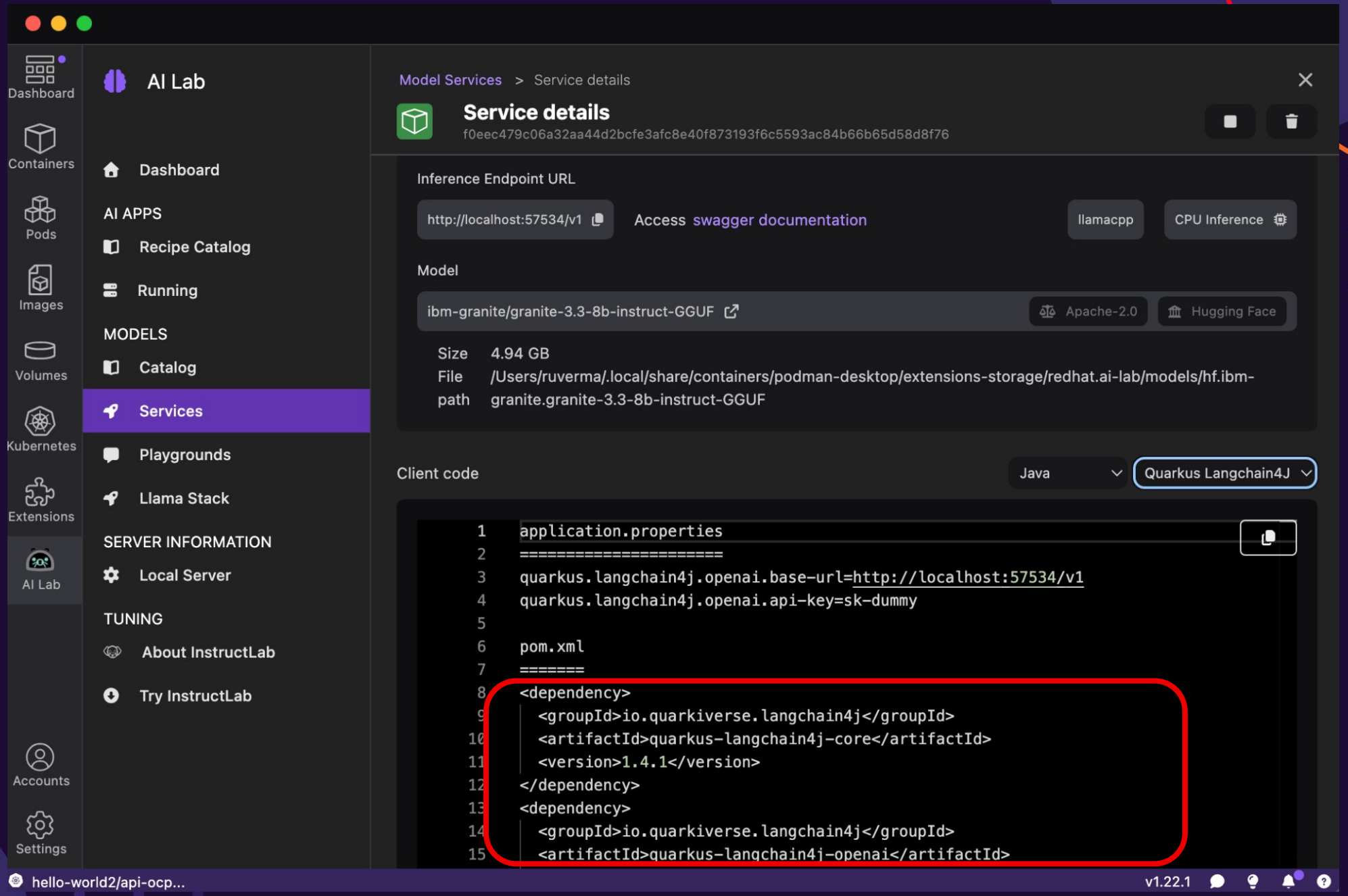
Podman Desktop


Dashboard
Containers
Pods
Images
Volumes
Kubernetes
Extensions
AI Lab
Accounts
Settings
AI Lab
Dashboard
AI APPS
Recipe Catalog
Running
MODELS
Catalog
Services
Playgrounds
Llama Stack
SERVER INFORMATION
Local Server
TUNING
About InstructLab
Try InstructLab
Models
Import
All
Downloaded
Imported
Available
STATUS
NAME
SIZE
AGE
ACTIONS
ibm-granite/granite-3.3-8b-ins...
Hugging Face - Apache-2.0
4.94 GB
7 hours
RAM usage: 4.6 GiB
ggerganov/whisper.cpp
Hugging Face - Apache-2.0
487.6 MB
3 hours
RAM usage: 464.45 MiB
facebook/detr-resnet-101
Hugging Face - Apache-2.0
242.98 M
3 hours
RAM usage: 231.72 MiB
microsoft/Phi-4-mini-reasonin...
Hugging Face - MIT
2.49 GB
3 hours
RAM usage: 2.31 GiB
mistralai/Mistral-Small-3.2-24B...
- Apache-2.0
RAM usage: 13.32 GiB
openai/gtp-oss-20b (Unsloth q...
Hugging Face - Apache-2.0
RAM usage: 10.8 GiB
qwen/qwen3-4b-GGUF
Hugging Face - Apache-2.0
RAM usage: N/A
qwen/Qwen3-4B-Thinking-250...
Hugging Face - Apache-2.0
RAM usage: N/A
ibm-granite/granite-4.0-tiny-G...
Hugging Face - Apache-2.0
RAM usage: 3.93 GiB
hello-world2/api-ocp...
v1.22.1


Dashboard
Containers
Pods
Images
Volumes
Kubernetes
Extensions
AI Lab
Accounts
Settings
AI Lab
Dashboard
AI APPS
Recipe Catalog
Running
MODELS
Catalog
Services
Playgrounds
Llama Stack
SERVER INFORMATION
Local Server
TUNING
About InstructLab
Try InstructLab
Model Services > Creating Model service
Creating Model service
Model
ibm-granite/granite-3.3-8b-instruct-GGUF
Container port
62082
Create service
hello-world2/api-ocp...
v1.22.1


Dashboard
Containers
Pods
Images
Volumes
Kubernetes
Extensions
AI Lab
Accounts
Settings
AI Lab
Dashboard
AI APPS
Recipe Catalog
Running
MODELS
Catalog
Services
Playgrounds
Llama Stack
SERVER INFORMATION
Local Server
TUNING
About InstructLab
Try InstructLab
Model Services > Service details
Service details
68e7f6462181c7926dc0b1daeab73937577dfb271070bc1cdc8c5bab6ec76869
Inference Endpoint URL
http://localhost:62082/v1
Access swagger documentation
llamacpp
GPU Inference
Model
ibm-granite/granite-3.3-8b-instruct-GGUF
Apache-2.0
Hugging Face
File path
/Users/ruverma/.local/share/containers/podman-desktop/extensions-storage/redhat.ai-lab/models/hf.ibm-granite.granite-3.3-8b-instruct-GGUF
Client code
cURL
cURL
curl --location 'http://localhost:62082/v1/chat/completions' \
--header 'Content-Type: application/json' \
--data '{
"messages": [
{
"content": "You are a helpful assistant.",
"role": "system"
},
{
"content": "What is the capital of France?",
"role": "user"
}
]
}'
hello-world2/api-ocp...
v1.22.1


Dashboard
Containers
Pods
Images
Volumes
Kubernetes
Extensions
AI Lab
Accounts
Settings
AI Lab
Dashboard
AI APPS
Recipe Catalog
Running
MODELS
Catalog
Services
Playgrounds
Llama Stack
SERVER INFORMATION
Local Server
TUNING
About InstructLab
Try InstructLab
Model Services > Service details
Service details
f0eec479c06a32aa44d2bcfe3afc8e40f873193f6c5593ac84b66b65d58d8f76
Inference Endpoint URL
http://localhost:57534/v1
Access swagger documentation
llamacpp
CPU Inference
Model
ibm-granite/granite-3.3-8b-instruct-GGUF
Apache-2.0
Hugging Face
Size 4.94 GB
File path /Users/ruverma/.local/share/containers/podman-desktop/extensions-storage/redhat.ai-lab/models/hf.ibm-granite.granite-3.3-8b-instruct-GGUF
Client code
Java
Quarkus Langchain4J
application.properties
quarkus.langchain4j.openai.base-url=http://localhost:57534/v1
quarkus.langchain4j.openai.api-key=sk-dummy
pom.xml
<dependency>
<groupId>io.quarkiverse.langchain4j</groupId>
<artifactId>quarkus-langchain4j-core</artifactId>
<version>1.4.1</version>
</dependency>
<dependency>
<groupId>io.quarkiverse.langchain4j</groupId>
<artifactId>quarkus-langchain4j-openai</artifactId>
hello-world2/api-ocp...
v1.22.1


Dashboard
Containers
Pods
Images
Volumes
Kubernetes
Extensions
AI Lab
Accounts
Settings
AI Lab
Dashboard
AI APPS
Recipe Catalog
Running
MODELS
Catalog
Services
Playgrounds
Llama Stack
SERVER INFORMATION
Local Server
TUNING
About InstructLab
Try InstructLab
Model Services > Service details
Service details
f0eec479c06a32aa44d2bcfe3afc8e40f873193f6c5593ac84b66b65d58d8f76
Inference Endpoint URL
http://localhost:57534/v1
Access swagger documentation
llamacpp
CPU Inference
Model
ibm-granite/granite-3.3-8b-instruct-GGUF
Apache-2.0
Hugging Face
Size 4.94 GB
File path /Users/ruverma/.local/share/containers/podman-desktop/extensions-storage/redhat.ai-lab/models/hf.ibm-granite.granite-3.3-8b-instruct-GGUF
Client code
Java
Quarkus Langchain4J
application.properties
quarkus.langchain4j.openai.base-url=http://localhost:57534/v1
quarkus.langchain4j.openai.api-key=sk-dummy
pom.xml
<dependency>
<groupId>io.quarkiverse.langchain4j</groupId>
<artifactId>quarkus-langchain4j-core</artifactId>
<version>1.4.1</version>
</dependency>
<dependency>
<groupId>io.quarkiverse.langchain4j</groupId>
<artifactId>quarkus-langchain4j-openai</artifactId>
hello-world2/api-ocp...
v1.22.1

Schritt 1: Quarkus vorbereiten

Frontend

REST-API

Quarkus

```java
package dev.langchain4j.quarkus.workshop;

import dev.langchain4j.service.SystemMessage;
import io.quarkiverse.langchain4j.
RegisterAiService;
import jakarta.enterprise.context.SessionScoped;

@SessionScoped
@RegisterAiService
public interface CustomerSupportAgent {

    @SystemMessage("You are a helpful agent")
    String chat(String userMessage);
}
```

Backend

LangChain4j

Podman Desktop

Schritt 1: Quarkus vorbereiten

- Dependency Management durch Maven
- Dependencies aus Podman AI in die `pom.xml` einfügen
- Maven lädt alle benötigten Bibliotheken runter

```
<?xml version="1.0"?>
<project ...>

    <!-- ... basic Quarkus configuration ... -->

    <dependencyManagement>
        <dependencies>
            <!-- Quarkus BOM -->
            <dependency>
                <groupId>io.quarkus.platform</groupId>
                <artifactId>quarkus-bom</artifactId>
                <version>${quarkus.platform.version}
</version>       <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>io.quarkus</groupId>
            <artifactId>quarkus-rest</artifactId>
        </dependency>
    </dependencies>

    <!-- ... -->

</project>
```

Quarkus vorbereiten

```xml
<?xml version="1.0"?>
<project ...>

    <!-- ... basic Quarkus configuration ... -->

    <dependencyManagement>
        <dependencies>
            <!-- Quarkus BOM -->
            <dependency>
                <groupId>io.quarkus.platform</groupId>
                <artifactId>quarkus-bom</artifactId>
                <version>${quarkus.platform.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>

            <!-- ⭐ Added: LangChain4j BOM -->
            <dependency>
                <groupId>io.quarkiverse.langchain4j</groupId>
                <artifactId>quarkus-langchain4j-bom</artifactId>
                <version>${quarkus-langchain4j.version}
</version>       <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

        <!-- ⭐ Added: LangChain4j OpenAI -->
        <dependency>
            <groupId>io.quarkiverse.langchain4j</groupId>
            <artifactId>quarkus-langchain4j-openai</artifactId>
        </dependency>
    </dependencies>

    <!-- ... -->
```

Schritt 2: Konfiguration

- application.properties zentrale Datei für Konfigurationen der Quarkus-Anwendung
- Einstellungen für: LLM-Provider, Ports, Logging, Datenbanken etc.
- Einfacher Schlüssel-Wert-Aufbau: key=value

application.properties

```
quarkus.langchain4j.openai.api-key={OPENAI-KEY}

quarkus.langchain4j.openai.chat-model.model-name=gpt-4o
quarkus.langchain4j.openai.chat-model.log-requests=true
quarkus.langchain4j.openai.chat-model.log-responses=true

quarkus.langchain4j.openai.chat-model.temperature=1.0
quarkus.langchain4j.openai.chat-model.max-tokens=1000
quarkus.langchain4j.openai.chat-model.frequency-penalty=0
```

Schritt 3: LLM-Service implementieren

Quarkus

REST-API

```java
package dev.langchain4j.quarkus.workshop;

import dev.langchain4j.service.SystemMessage;
import io.quarkiverse.langchain4j.
RegisterAiService;
import jakarta.enterprise.context.SessionScoped;

@SessionScoped
@RegisterAiService
public interface CustomerSupportAgent {

    @SystemMessage("You are a helpful agent")
    String chat(String userMessage);
}
```

LangChain4j

Frontend

Backend

Schritt 3: LLM-Service implementieren

- ➔ Annotationen reduzieren unnötigen Code (@RegisterAiService)
- ➔ LLM-Ausgaben werden auf Java-Objekte abgebildet
- ➔ Bisherige Konzepte werden abstrahiert und integriert (Http handling, Json Handling)

Ohne Annotationen

Ohne @RegisterAiService

```
1 package dev.langchain4j.quarkus.workshop;
2
3 import jakarta.enterprise.context.SessionScoped;
4
5 @SessionScoped
6 public interface CustomerSupportAgent {
7
8     String chat(String topic, String style);
9 }
10
```

Ohne @RegisterAiService

```java
import jakarta.enterprise.context.ApplicationScoped;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

@ApplicationScoped
public class CustomerSupportAgentImpl implements CustomerSupportAgent {

    private static final String API_KEY = System.getenv("OPENAI_API_KEY");
    private final HttpClient httpClient = HttpClient.newHttpClient();

    @Override
    public String chat(String topic, String style) {
        try {
            // 1. Manually construct the complex JSON payload
            String jsonPayload = String.format("""
                {
                    "model": "gpt-3.5-turbo",
                    "messages": [
                        {"role": "system", "content": "You are a support Agent."},
                        {"role": "user", "content": "I want to book a car"}
                    ],
                    "temperature": 0.7
                }
                """, topic, style);

            // 2. Manually build the HTTP request with headers
            HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("https://api.openai.com/v1/chat/completions"))
                .header("Content-Type", "application/json")
                .header("Authorization", "Bearer " + API_KEY) // Authentication
                .POST(HttpRequest.BodyPublishers.ofString(jsonPayload))
                .build();

        }
    }
    // ... helper method to parse JSON ...
```

Mit @RegisterAiService

```
 1 package dev.langchain4j.quarkus.workshop;
 2
 3 import dev.langchain4j.service.SystemMessage;
 4 import io.quarkiverse.langchain4j.RegisterAiService;
 5 import jakarta.enterprise.context.SessionScoped;
 6
 7 @SessionScoped
 8 @RegisterAiService
 9 public interface CustomerSupportAgent {
10
11     @SystemMessage("You are a helpful agent")
12     String chat(String topic, String style);
13 }
14
```

mit @RegisterAiService

```java
package dev.langchain4j.quarkus.workshop;

import io.quarkus.websockets.next.OnOpen;
import io.quarkus.websockets.next.OnTextMessage;
import io.quarkus.websockets.next.WebSocket;

@WebSocket(path = "/customer-support-agent")
public class CustomerSupportAgentWebSocket {

    private final CustomerSupportAgent customerSupportAgent;

    public CustomerSupportAgentWebSocket(CustomerSupportAgent customerSupportAgent) {
        this.customerSupportAgent = customerSupportAgent;
    }

    @OnTextMessage
    public String onTextMessage(String message, String tone) {
        return customerSupportAgent.chat(message,tone);
    }
}
```

Demo time


demo god
PLEASE WORK
QUARKUS
QUARKUS
QUARKUS
LANGCHAIN
LANGCHAIN
LANGCHAIN
LANGCHAIN

Conclusion

Ausblick: Enterprise Ready

Frontend

REST-API

Quarkus

```java
package dev.langchain4j.quarkus.workshop;

import dev.langchain4j.service.SystemMessage;
import io.quarkiverse.langchain4j.
RegisterAiService;
import jakarta.enterprise.context.SessionScoped;

@SessionScoped
@RegisterAiService
public interface CustomerSupportAgent {

    @SystemMessage("You are a helpful agent")
    String chat(String userMessage);
}
```

Backend

LangChain4j

Podman Desktop

Ausblick: Enterprise Ready

Guardrailing, Observability, MCP

Frontend

REST-API

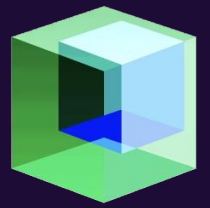
Red Hat build of Quarkus

```java
package dev.langchain4j.quarkus.workshop;

import dev.langchain4j.service.SystemMessage;
import io.quarkiverse.langchain4j.
RegisterAiService;
import jakarta.enterprise.context.SessionScoped;

@SessionScoped
@RegisterAiService
public interface CustomerSupportAgent {

    @SystemMessage("You are a helpful agent")
    String chat(String userMessage);
}
```

Backend

LangChain4j

RAG



Red Hat OpenShift AI

Red Hat Summit

Connect

Thank you


linkedin.com/company/red-hat


facebook.com/redhatinc


youtube.com/user/RedHatVideos


twitter.com/RedHat

Jetzt Session bewerten!

Einfach QR-Code scannen, Session aus der Liste wählen und bewerten. **Vielen Dank!**

red.ht/rhsc-darmstadt-feedback