The Movie Chat: End-to-End Cloud-Native AI Application

By - Devesh Ruttala (49385)

1. Project Overview

"The Movie Chat" is a Generative AI application designed to bridge the gap between natural language and relational databases. Unlike traditional chatbots that rely on static text or vector stores (RAG), this application utilizes a **Text-to-SQL** architecture.

When a user asks a question (e.g., *"Who acted in Inception?"*), the system does not "guess" the answer. Instead, it:

1. Interprets the user's intent using OpenAI (GPT-3.5/4).
2. Generates a precise SQL query based on the database schema.
3. Executes the query against a PostgreSQL database.
4. Returns the exact, factual data to the user.

2. Technology Stack

Backend & AI

- **Language:** Java 21
- **Framework:** Spring Boot 3.3.0
- **AI Integration:** Spring AI (0.8.1 / 1.0.0-M1)
- **LLM Provider:** OpenAI API (GPT-3.5 Turbo)
- **Build Tool:** Apache Maven

Database

- **Engine:** PostgreSQL 16
- **ORM:** Spring Data JPA / Hibernate
- **Migration:** data.sql (Schema initialization & seeding)

Infrastructure & Cloud

- **Containerization:** Docker
- **Orchestration:** Kubernetes (GKE - Google Kubernetes Engine)
- **Registry:** Google Artifact Registry (GAR)
- **Infrastructure Code:** Kubernetes YAML Manifests

Automation

- **Language:** Python 3
- **Tool:** Custom Gemini Agent Wrapper (agent_tool.py)

3. Project Architecture Description

A. Backend Application (`src/main/java`)

The core application is built using **Spring Boot 3.3** and follows the **Model-View-Controller (MVC)** design pattern.

- **Controller Layer:** `ChatController.java` exposes the `/api/chat` REST endpoint. It accepts user questions as JSON and returns the answer.
- **Service Layer:** `ChatService.java` contains the business logic. It handles the "Dual-Mode" intelligence:
    - *Real Mode:* Uses **Spring AI** to send the database schema and user question to OpenAI, receives a generated SQL query, validates it, and executes it via `JdbcTemplate`.
    - *Mock Mode:* Acts as a fallback if no API key is detected, performing keyword searches to ensure the application remains testable without costs.
- **Data Layer:** The `entity` package defines the JPA (Java Persistence API) objects that map Java classes directly to PostgreSQL database tables.

B. Resources & Configuration (`src/main/resources`)

- **`application.properties`:** The central configuration file. It manages database connection strings (`jdbc:postgresql://...`), Hibernate settings (Dialects), and API keys.

- **`data.sql`:** A SQL script that runs automatically on application startup (configured via `spring.sql.init.mode=always`). It populates the database with initial movie and artist data so the chatbot has content to query immediately.
- **`static/index.html`:** A lightweight frontend interface served directly by Spring Boot, allowing users to interact with the API via a web browser.

C. Cloud Infrastructure (`k8s/`)

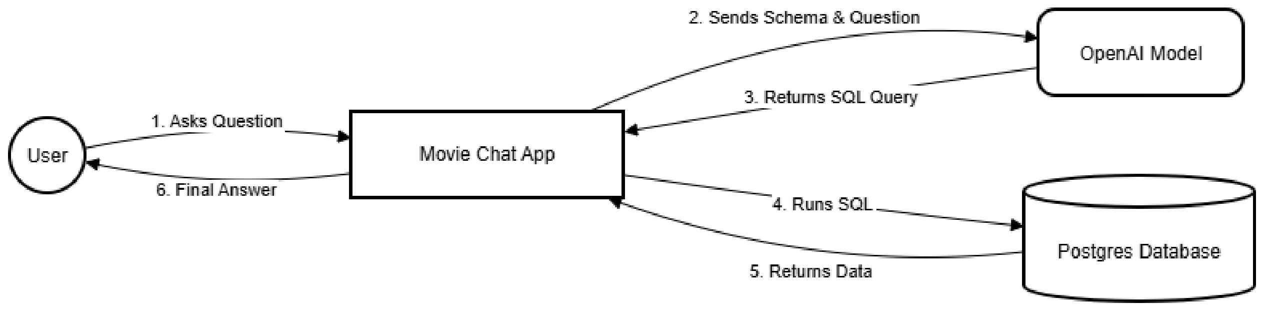
This directory contains "Infrastructure as Code" (IaC) to deploy the application to Google Kubernetes Engine (GKE).

- **`postgres.yaml`:** Deploys a stateful PostgreSQL instance inside the cluster.
- **`deployment.yaml`:** Defines the Movie Chat application pod. It specifies the Docker image source (Google Artifact Registry) and injects sensitive environment variables (API Keys, DB Credentials) at runtime.
- **`service.yaml`:** Defines a Kubernetes `LoadBalancer` service, which provisions an external static IP address to make the application accessible via the public internet.

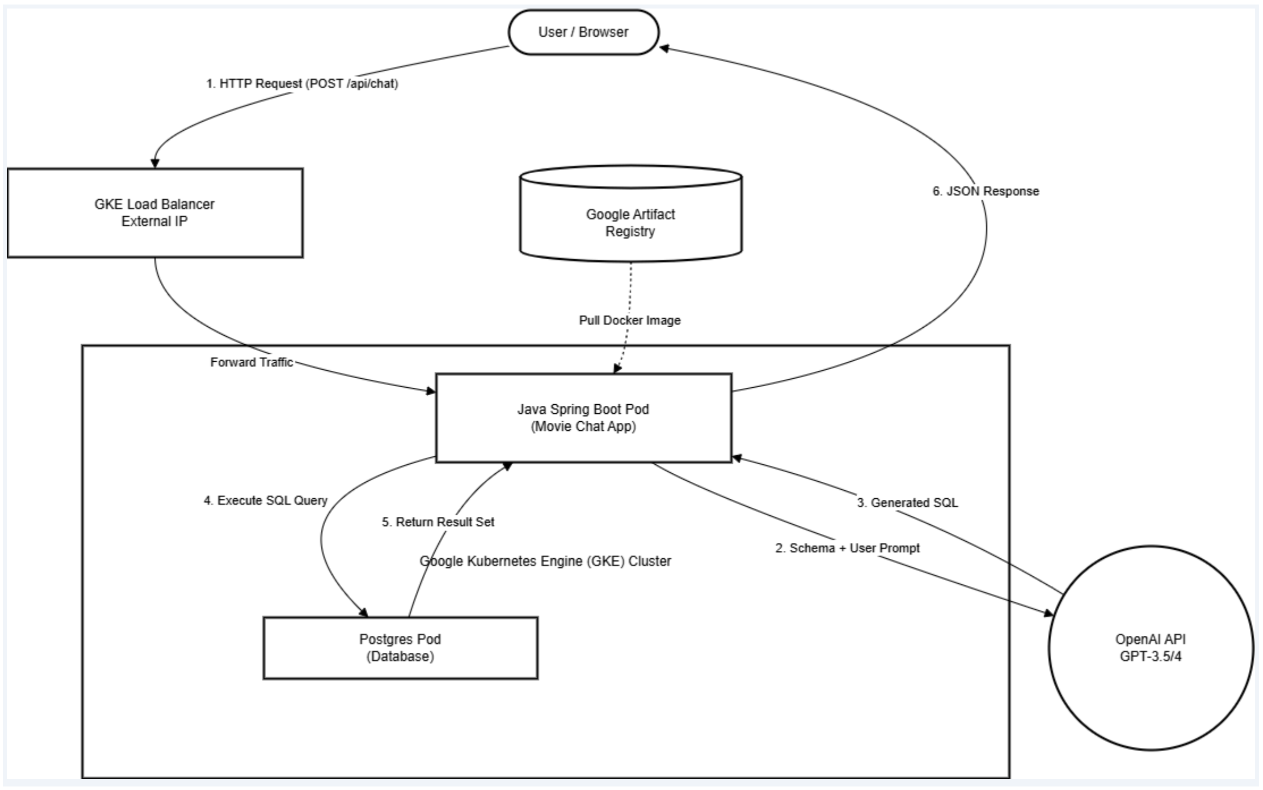
D. Automation & DevOps

- **`Dockerfile`:** A multi-stage build script. It first compiles the Java application using Maven and then packages the resulting JAR file into a lightweight Eclipse Temurin JDK image for production.
- **`agent_tool.py`:** A custom Python script designed to be triggered by a Gemini Agent. It automates the entire CI/CD pipeline:
    1. Builds the Docker image.
    2. Pushes the image to Google Artifact Registry.
    3. Applies Kubernetes manifests (`kubectl apply`).
    4. Triggers a rollout restart to update the application live.

Local Architecture

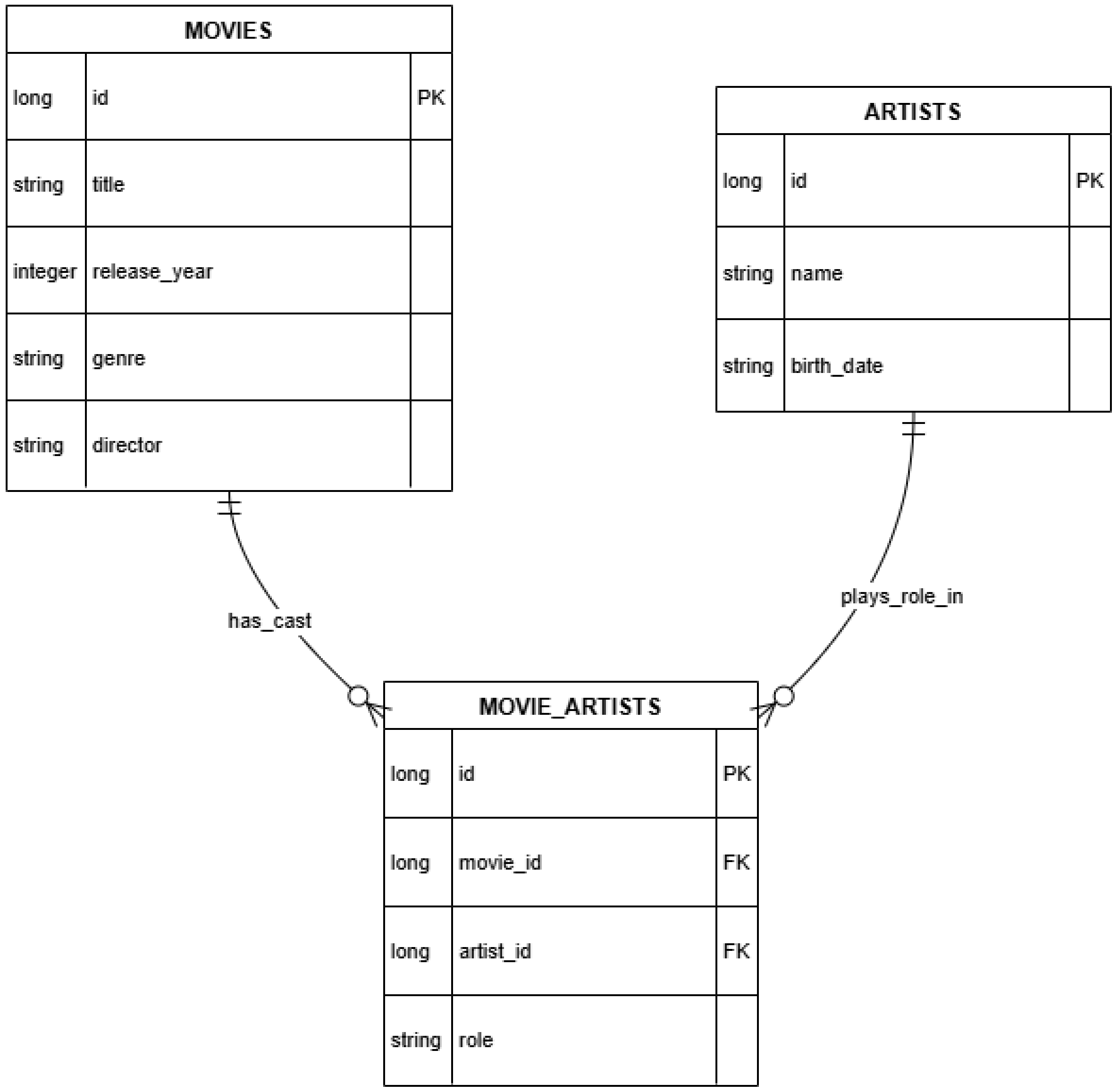


GKE Deploment Architecture



4. Database Schema

The application uses a normalized relational schema to handle complex relationships between movies and the people involved in them.



- **Movies:** id, title, release_year, genre, director
- **Artists:** id, name, birth_date
- **Movie_Artists (Join Table):** Links Movies and Artists with a specific role (e.g., Actor, Producer).

5. Phase 1: Local Development Setup

Prerequisites

- Java JDK 21
- Docker Desktop (Running)
- Maven
- OpenAI API Key (Optional for Mock Mode, Required for AI Mode)

Step-by-Step Local Run

1. **Clone the Repository:**
   ```bash
   git clone <repository-url>
   cd moviechat
   ```

2. Configure Environment:
   Open src/main/resources/application.properties.
   - Set spring.ai.openai.api-key to your key, or leave as placeholder to trigger "Mock Mode".
   - Ensure spring.datasource.url points to jdbc:postgresql://127.0.0.1:5433/moviedb.
3. Start Local Database:
   We use Docker to run Postgres locally to avoid installing it on Windows directly.
   ```bash
   docker run --name movie_chat_db -e POSTGRES_DB=moviedb -e
   POSTGRES_USER=admin -e POSTGRES_PASSWORD=password -p 5433:5432 -d
   postgres:16
   ```

4. **Run the Application:**
   ```bash
   mvn spring-boot:run
   ```

5. Access:

Open http://localhost:8080 in your browser.

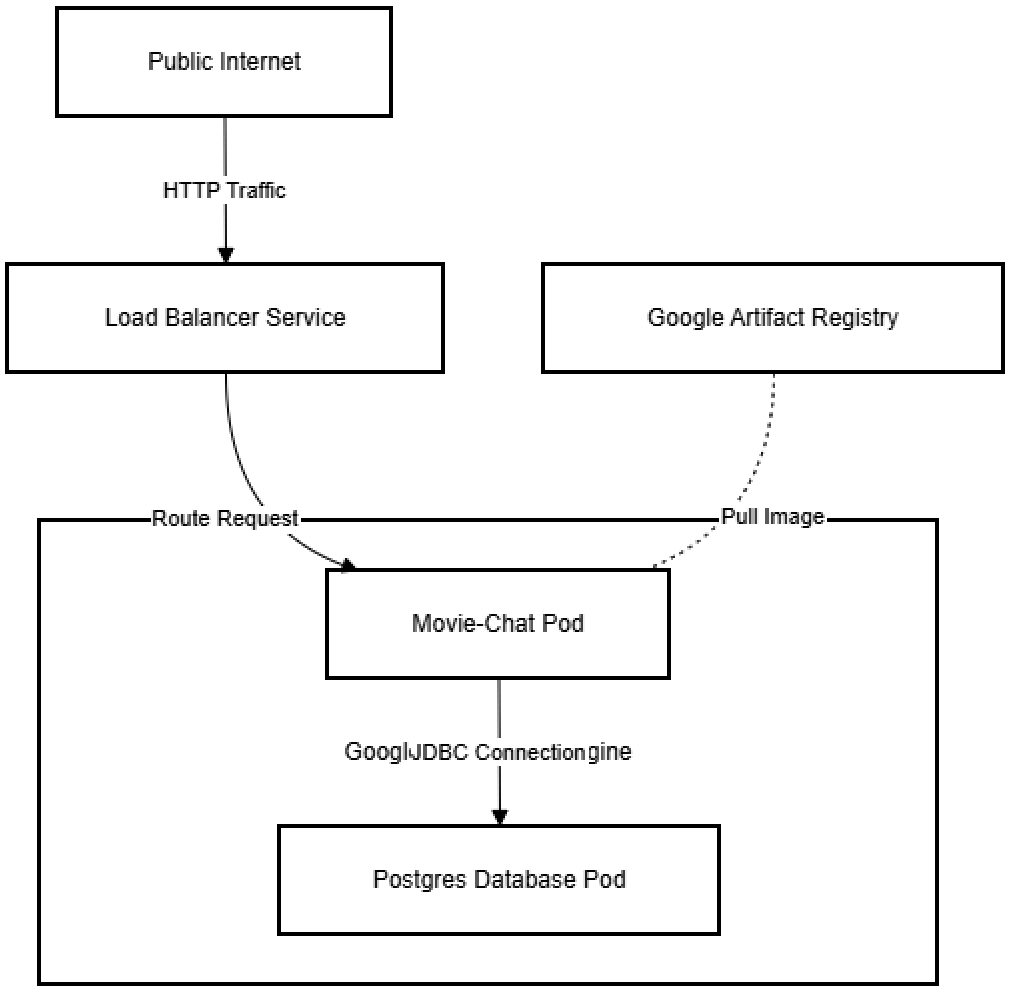
Key Feature: Dual-Mode Service

The ChatService.java contains logic to detect if a valid API key is present.

- **Real AI Mode:** Converts natural language to SQL using OpenAI.
- **Mock Mode:** If no key is found, it performs a keyword search (e.g., searching for "Matrix" returns Matrix data) to allow UI/DB testing without costs.

6. Phase 2 & 3: Cloud Infrastructure (GKE)

The project is designed to be "Cloud Native". It does not rely on local files but runs entirely within containers managed by Kubernetes.



Infrastructure Components

1. **Google Artifact Registry:** Stores the Docker images securely.

2. **GKE Cluster:** A managed Kubernetes environment.
3. **Workloads:**
   - **Postgres Pod:** Runs the database inside the cluster (Stateful).
   - **Movie-Chat Pod:** The Java application (Stateless).
4. **Service (LoadBalancer):** Exposes the Movie-Chat pod to the public internet via an external IP.

Configuration Files (k8s/)

- **deployment.yaml:** Defines the Java app replica set, environment variables (API Keys, DB Credentials), and the image source from Artifact Registry.
- **postgres.yaml:** Defines a single-replica Postgres instance inside the cluster.
- **service.yaml:** Maps Port 80 (Public) to Port 8080 (Container).

7. Phase 4: Automated Deployment (The Agent Tool)

To solve the challenge of manual deployments, a Python automation tool (agent_tool.py) was created. This script acts as a CI/CD pipeline that a Gemini Agent could trigger.

What the Script Does:

1. **Environment Check:** Verifies Docker and Gcloud are installed.
2. **Build:** Runs docker build to create a new image from the Java source.
3. **Auth:** Authenticates Docker with Google Cloud.
4. **Push:** Uploads the new image to Google Artifact Registry.
5. **Deploy DB:** Applies the Postgres configuration to Kubernetes.
6. **Deploy App:** Applies the Java App configuration.
7. **Rollout:** Forces a restart of the pods to pull the latest code.

How to Run Deployment

Ensure you have gcloud authenticated (gcloud auth login). Then run:

Bash

```
python agent_tool.py
```

8. API Documentation

The backend exposes a single RESTful endpoint for the frontend to consume.

Chat Endpoint

- **URL:** /api/chat
- **Method:** POST
- **Content-Type:** application/json

Request Body:

JSON

```
{
  "question": "Who directed Interstellar?"
}
```

Response Body (Success):

JSON

```json
{
  "query": "Who directed Interstellar?",
  "sql": "SELECT director FROM movies WHERE title ILIKE '%Interstellar%'",
  "answer": [
    {
      "director": "Christopher Nolan"
    }
  ]
}
```

9. Troubleshooting & Common Fixes

During development, several environment-specific challenges were solved:

1. **Docker on Windows (EOF Error):**
   - *Issue:* Maven build crashing Docker due to memory/connection.
   - *Fix:* Optimized Dockerfile to cache dependencies (mvn dependency:go-offline) and ensured Docker Desktop was restarted.
2. **Postgres "Unable to determine Dialect":**
   - *Issue:* Hibernate could not handshake with the DB on startup.
   - Fix: Hardcoded the dialect in application.properties: spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect.
3. **"TimeZone: Asia/Calcutta" Error:**
   - *Issue:* Java sent a timezone string that Postgres didn't recognize.
   - Fix: Forced UTC timezone in MoviechatApplication.java: TimeZone.setDefault(TimeZone.getTimeZone("UTC"));.
4. **Empty Database Responses:**
   - *Issue:* data.sql ran before tables were created.
   - *Fix:* Added spring.jpa.defer-datasource-initialization=true and spring.sql.init.mode=always to ensure data seeding happens after table creation.
5. **GKE Authentication:**
   - *Issue:* kubectl could not connect to the cluster.

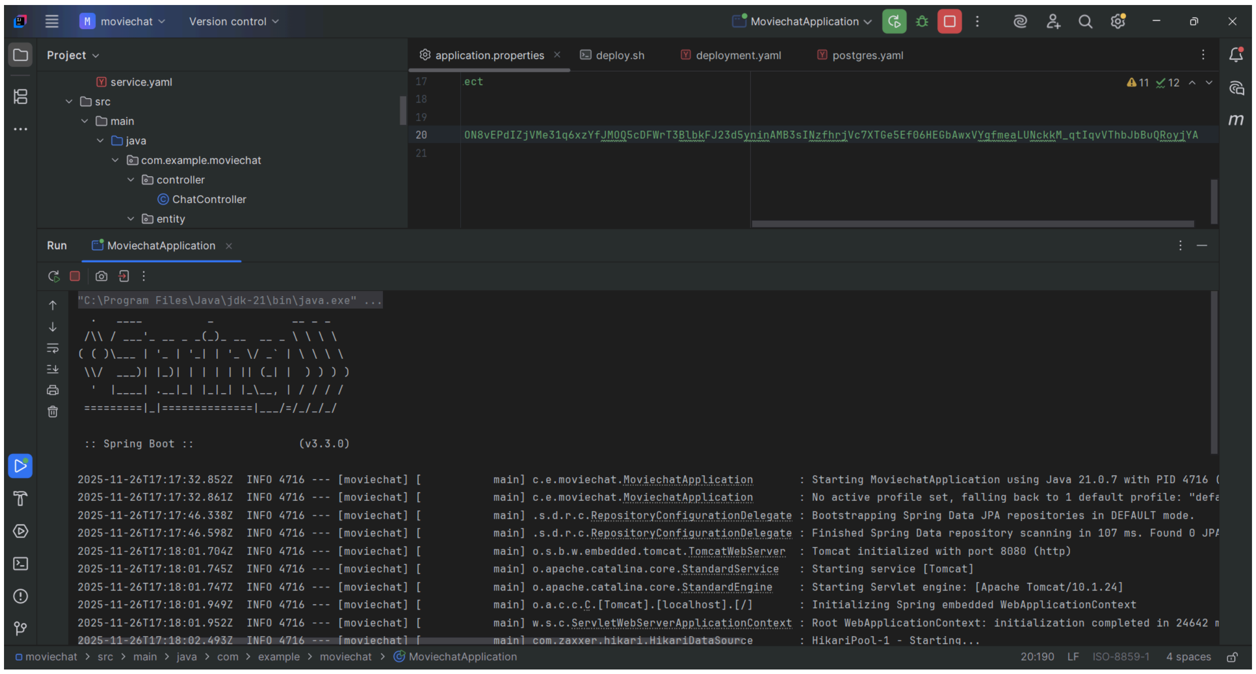
◦ *Fix:* Installed the auth plugin: gcloud components install gke-gcloud-auth-plugin.

10. Future Improvements

- **Cloud SQL:** Migration from in-cluster Postgres to managed Google Cloud SQL for better durability.
- **RAG Integration:** Add a Vector Store to answer questions about movie *plots* (unstructured text) alongside the SQL data.
- **React Frontend:** Replace the static HTML with a modern React or Angular interface.

11. Implementation Screenshots:

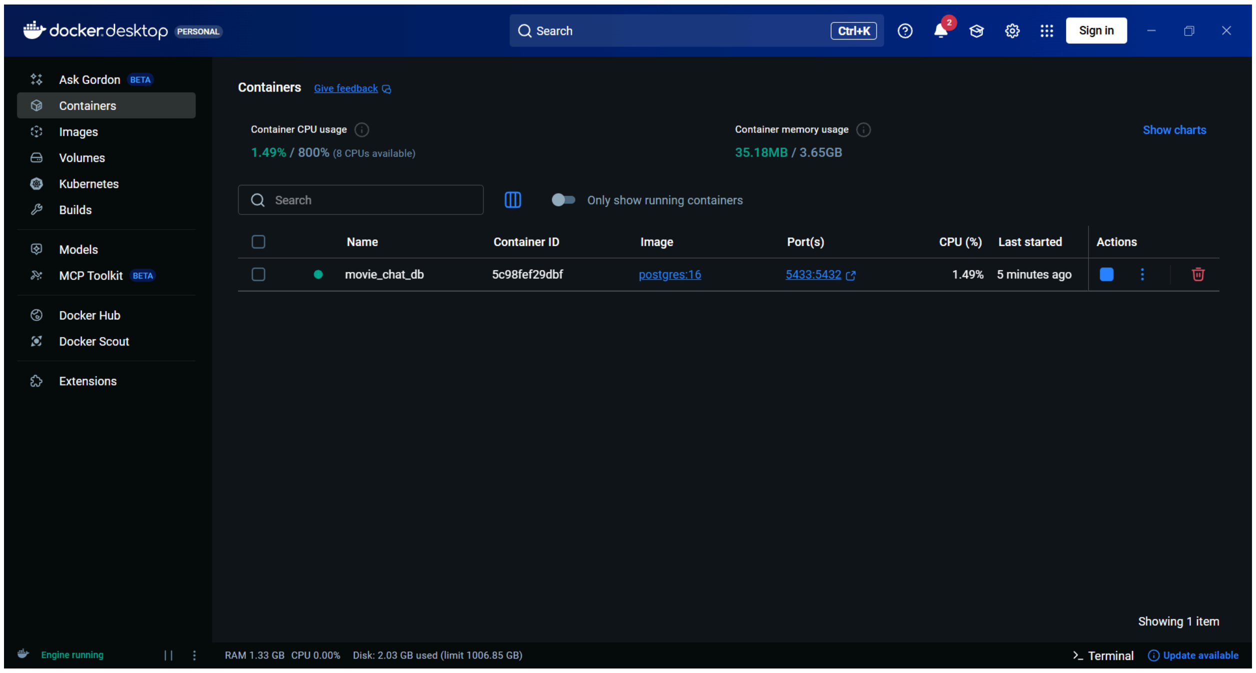
1. Local Development & Testing



```
ect


ON8vEPdIZjVMe31q6xzYfJMOQ5cDFWrT3BlbkFJ23d5yninAMB3sINzfhrjVc7XTGe5Ef06HEGbAwxVYgfmeaLUNckkM_qtIqvVThbJbBuQRoyjYA
```

```
"C:\Program Files\Java\jdk-21\bin\java.exe" ...
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.0)

2025-11-26T17:17:32.852Z  INFO 4716 --- [moviechat] [           main] c.e.moviechat.MoviechatApplication       : Starting MoviechatApplication using Java 21.0.7 with PID 4716 (
2025-11-26T17:17:32.861Z  INFO 4716 --- [moviechat] [           main] c.e.moviechat.MoviechatApplication       : No active profile set, falling back to 1 default profile: "defa
2025-11-26T17:17:46.338Z  INFO 4716 --- [moviechat] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-11-26T17:17:46.598Z  INFO 4716 --- [moviechat] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 107 ms. Found 0 JPA
2025-11-26T17:18:01.704Z  INFO 4716 --- [moviechat] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-11-26T17:18:01.745Z  INFO 4716 --- [moviechat] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-11-26T17:18:01.747Z  INFO 4716 --- [moviechat] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1.24]
2025-11-26T17:18:01.949Z  INFO 4716 --- [moviechat] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2025-11-26T17:18:01.952Z  INFO 4716 --- [moviechat] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 24642 m
2025-11-26T17:18:02.493Z  INFO 4716 --- [moviechat] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
```

(Spring Boot banner & startup logs) **Description:** *Initial startup of the Spring Boot application in IntelliJ IDEA. The logs confirm the application has successfully connected to the local Docker database and is running on port 8080.*



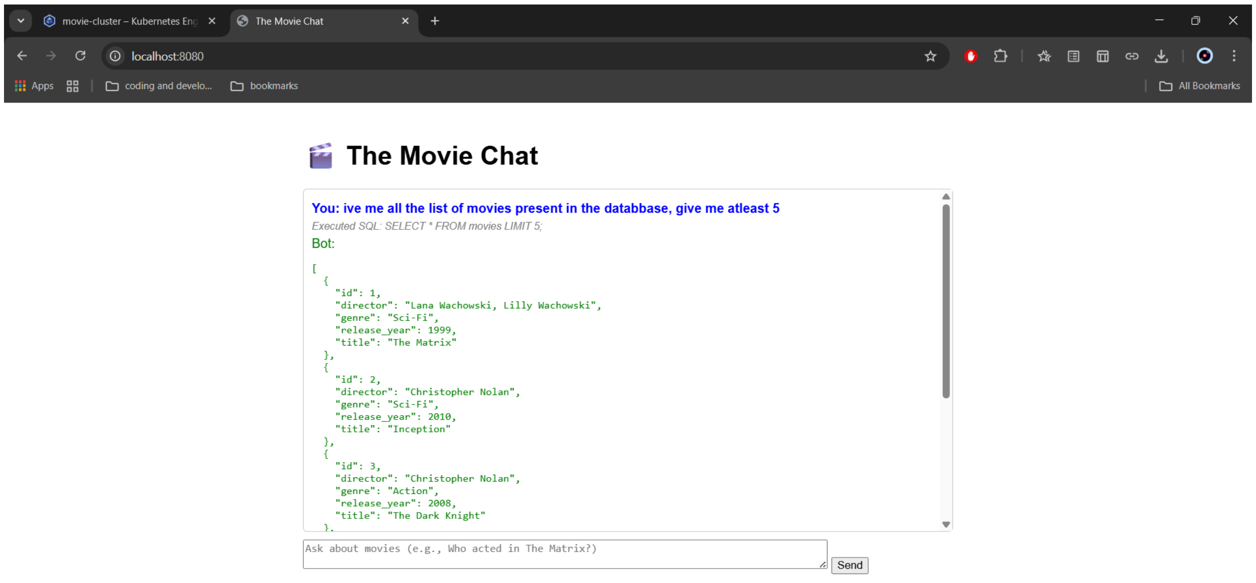
Container CPU usage
1.49% / 800% (8 CPUs available)

Container memory usage
35.18MB / 3.65GB

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
movie_chat_db	5c98fef29dbf	postgres:16	5433:5432	1.49%	5 minutes ago	

RAM 1.33 GB CPU 0.00% Disk: 2.03 GB used (limit 1006.85 GB)

(Docker Desktop Postgres container) **Description:** *Docker Desktop dashboard showing the local PostgreSQL container (`movie_chat_db`) running successfully. This container serves as the database for local development and testing.*



🎬 The Movie Chat

You: ive me all the list of movies present in the databbase, give me atleast 5

*Executed SQL: SELECT * FROM movies LIMIT 5;*

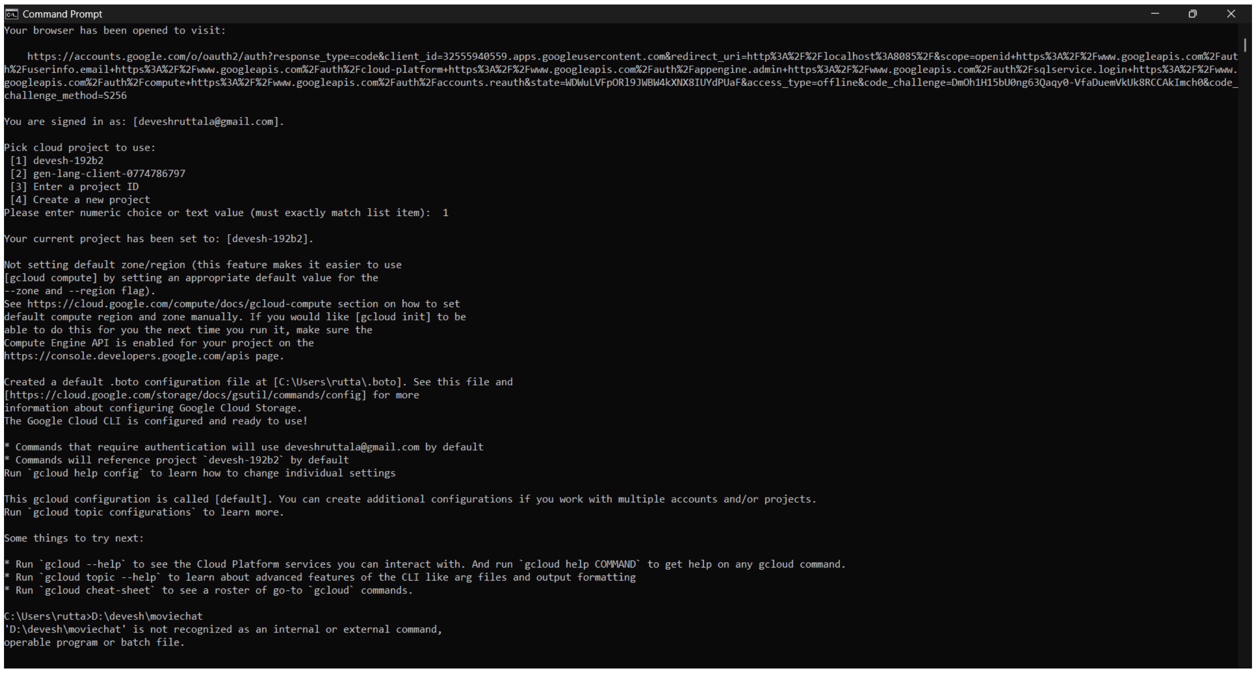
Bot:

```
[
  {
    "id": 1,
    "director": "Lana Wachowski, Lilly Wachowski",
    "genre": "Sci-Fi",
    "release_year": 1999,
    "title": "The Matrix"
  },
  {
    "id": 2,
    "director": "Christopher Nolan",
    "genre": "Sci-Fi",
    "release_year": 2010,
    "title": "Inception"
  },
  {
    "id": 3,
    "director": "Christopher Nolan",
    "genre": "Action",
    "release_year": 2008,
    "title": "The Dark Knight"
  },
```

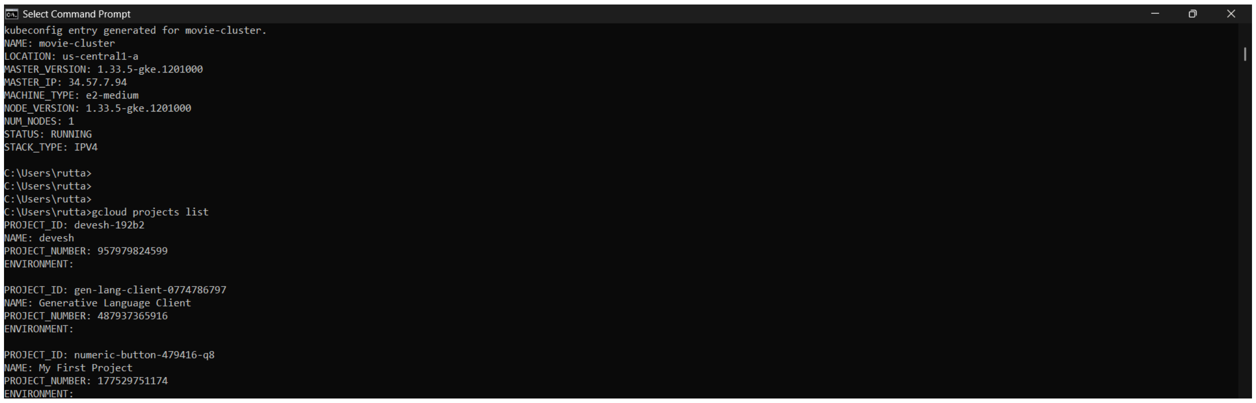
Ask about movies (e.g., Who acted in The Matrix?)

(Frontend with AI response - "Give me all the list of movies...") **Description:** *The web interface running locally. This screenshot demonstrates the Natural Language to SQL capability, where* *a user asks for a list of movies, and the bot correctly generates and executes the* `SELECT * FROM movies LIMIT 5;` *query.*

2. Cloud Infrastructure Setup (GKE)

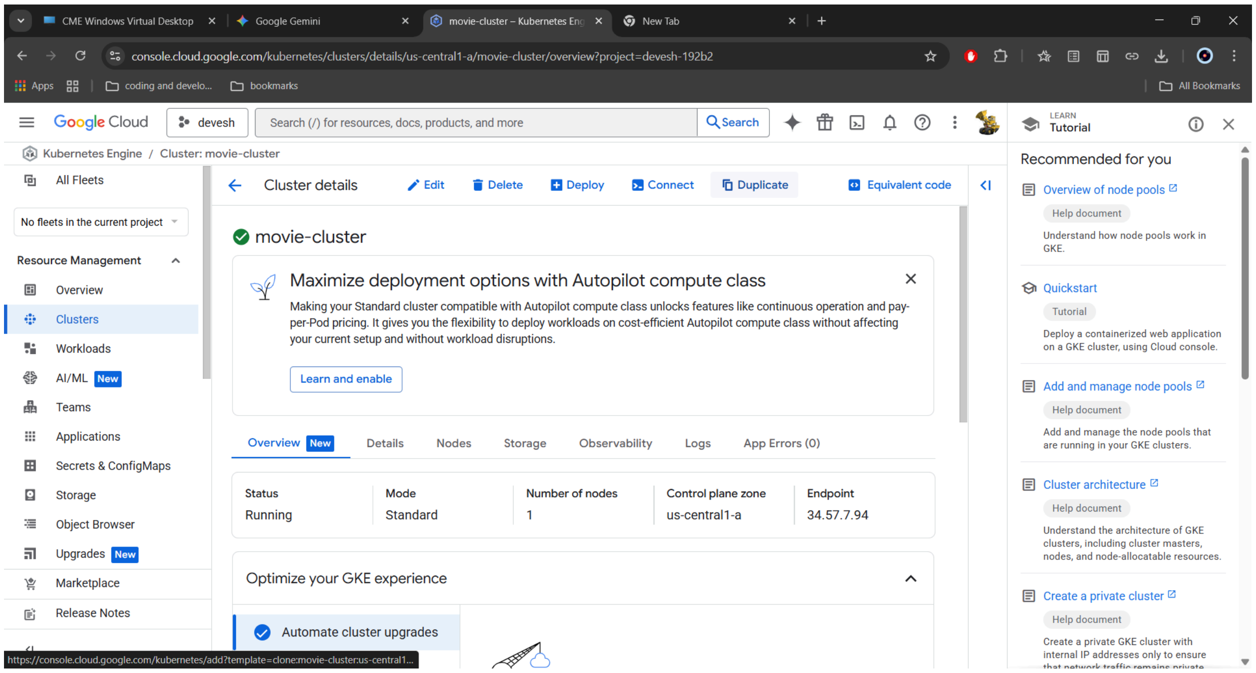


```
Your browser has been opened to visit:

    https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8085%2F&scope=openid+https%3A%2F%2Fwww.googleapis.com%2Faut
h%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fsqlservice.login+https%3A%2F%2Fwww.
googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Faccounts.reauth&state=WDWuLVFpORl9JWBW4kXNX8IUYdPUaF&access_type=offline&code_challenge=DmOh1H15bU0ng63Qaqy0-VfaDuemVkUk8RCCAkImch0&code_
challenge_method=S256

You are signed in as: [deveshruttala@gmail.com].

Pick cloud project to use:
 [1] devesh-192b2
 [2] gen-lang-client-0774786797
 [3] Enter a project ID
 [4] Create a new project
Please enter numeric choice or text value (must exactly match list item):  1

Your current project has been set to: [devesh-192b2].

Not setting default zone/region (this feature makes it easier to use
[gcloud compute] by setting an appropriate default value for the
--zone and --region flag).
See https://cloud.google.com/compute/docs/gcloud-compute section on how to set
default compute region and zone manually. If you would like [gcloud init] to be
able to do this for you the next time you run it, make sure the
Compute Engine API is enabled for your project on the
https://console.developers.google.com/apis page.

Created a default .boto configuration file at [C:\Users\rutta\.boto]. See this file and
[https://cloud.google.com/storage/docs/gsutil/commands/config] for more
information about configuring Google Cloud Storage.
The Google Cloud CLI is configured and ready to use!

* Commands that require authentication will use deveshruttala@gmail.com by default
* Commands will reference project `devesh-192b2` by default
Run `gcloud help config` to learn how to change individual settings

This gcloud configuration is called [default]. You can create additional configurations if you work with multiple accounts and/or projects.
Run `gcloud topic configurations` to learn more.

Some things to try next:

* Run `gcloud --help` to see the Cloud Platform services you can interact with. And run `gcloud help COMMAND` to get help on any gcloud command.
* Run `gcloud topic --help` to learn about advanced features of the CLI like arg files and output formatting
* Run `gcloud cheat-sheet` to see a roster of go-to `gcloud` commands.

C:\Users\rutta>D:\devesh\moviechat
'D:\devesh\moviechat' is not recognized as an internal or external command,
operable program or batch file.
```



```
kubeconfig entry generated for movie-cluster.
NAME: movie-cluster
LOCATION: us-central1-a
MASTER_VERSION: 1.33.5-gke.1201000
MASTER_IP: 34.57.7.94
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.33.5-gke.1201000
NUM_NODES: 1
STATUS: RUNNING
STACK_TYPE: IPV4

C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>gcloud projects list
PROJECT_ID: devesh-192b2
NAME: devesh
PROJECT_NUMBER: 957979824599
ENVIRONMENT:

PROJECT_ID: gen-lang-client-0774786797
NAME: Generative Language Client
PROJECT_NUMBER: 487937365916
ENVIRONMENT:

PROJECT_ID: numeric-button-479416-q8
NAME: My First Project
PROJECT_NUMBER: 177529751174
ENVIRONMENT:
```

(Terminal showing Project ID) **Description:** *Terminal output from the Google Cloud CLI (*`gcloud projects list`*), verifying the active Project ID (*`devesh-192b2`*) that will be used for the cloud deployment.*


CME Windows Virtual Desktop
Google Gemini
movie-cluster – Kubernetes Eng
New Tab
console.cloud.google.com/kubernetes/clusters/details/us-central1-a/movie-cluster/overview?project=devesh-192b2
Apps
coding and develo...
bookmarks
All Bookmarks
Google Cloud
devesh
Search (/) for resources, docs, products, and more
Search
Kubernetes Engine / Cluster: movie-cluster
All Fleets
No fleets in the current project
Resource Management
Overview
Clusters
Workloads
AI/ML New
Teams
Applications
Secrets & ConfigMaps
Storage
Object Browser
Upgrades New
Marketplace
Release Notes
Cluster details
Edit
Delete
Deploy
Connect
Duplicate
Equivalent code
LEARN
Tutorial
Recommended for you
https://console.cloud.google.com/kubernetes/add?template=clone:movie-cluster:us-central1...

movie-cluster

Maximize deployment options with Autopilot compute class

Making your Standard cluster compatible with Autopilot compute class unlocks features like continuous operation and pay-per-Pod pricing. It gives you the flexibility to deploy workloads on cost-efficient Autopilot compute class without affecting your current setup and without workload disruptions.

Learn and enable

Overview New | Details | Nodes | Storage | Observability | Logs | App Errors (0)

Status	Mode	Number of nodes	Control plane zone	Endpoint
Running	Standard	1	us-central1-a	34.57.7.94

Optimize your GKE experience

Automate cluster upgrades

Overview of node pools
Help document
Understand how node pools work in GKE.

Quickstart
Tutorial
Deploy a containerized web application on a GKE cluster, using Cloud console.

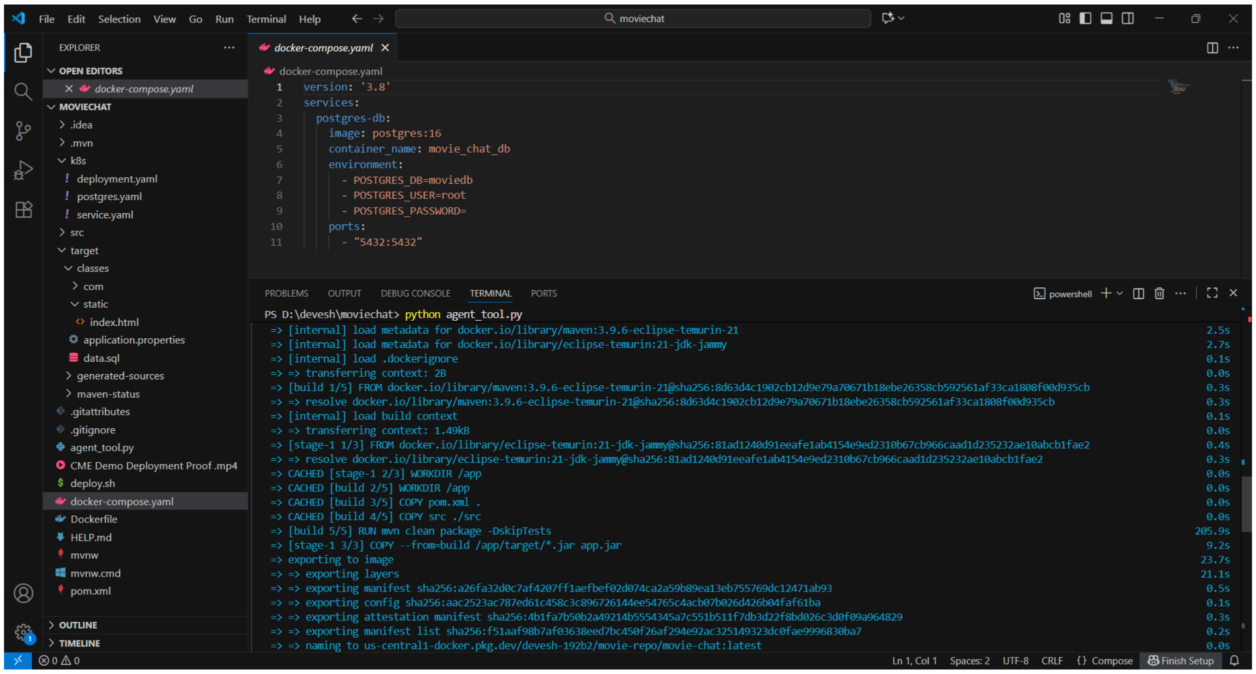
Add and manage node pools
Help document
Add and manage the node pools that are running in your GKE clusters.

Cluster architecture
Help document
Understand the architecture of GKE clusters, including cluster masters, nodes, and node-allocatable resources.

Create a private cluster
Help document
Create a private GKE cluster with internal IP addresses only to ensure that network traffic remains private

(Google Cloud Console Cluster Overview) **Description:** *The Google Cloud Console showing the Kubernetes cluster (`movie-cluster`) in a "Running" state. This is the managed environment where the application and database will be deployed.*

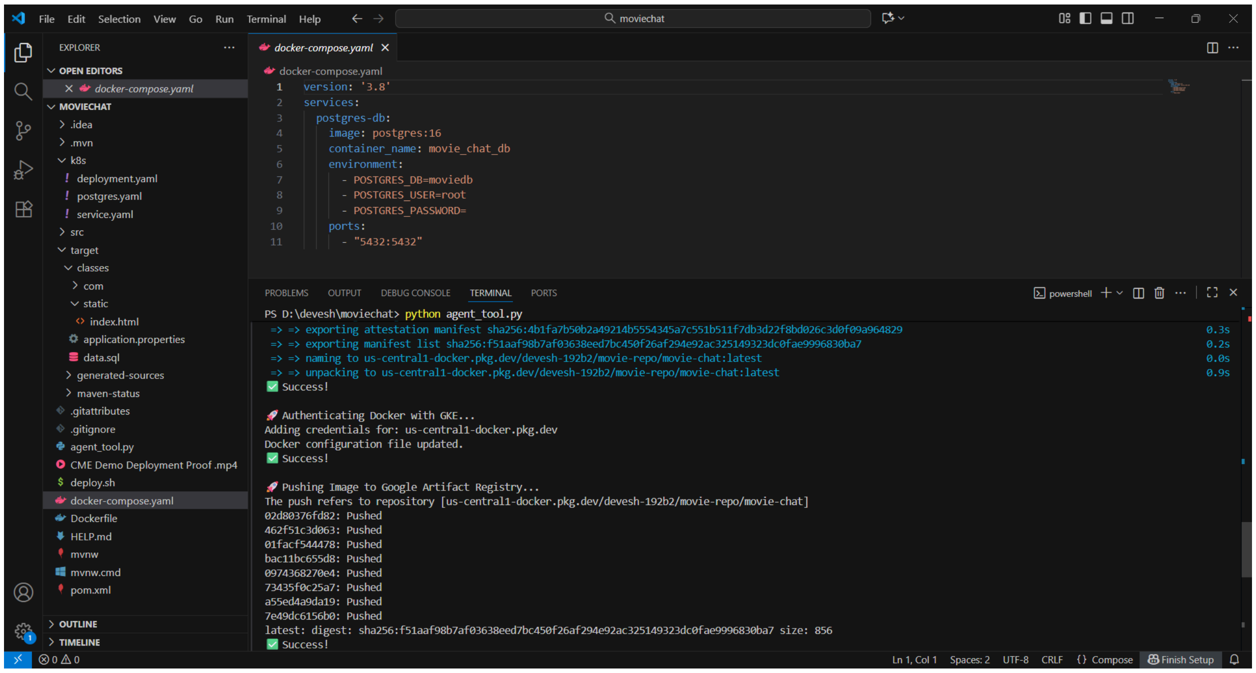
3. Automated Deployment



docker-compose.yaml

```yaml
version: '3.8'
services:
  postgres-db:
    image: postgres:16
    container_name: movie_chat_db
    environment:
      - POSTGRES_DB=moviedb
      - POSTGRES_USER=root
      - POSTGRES_PASSWORD=
    ports:
      - "5432:5432"
```

```
PS D:\devesh\moviechat> python agent_tool.py
 => [internal] load metadata for docker.io/library/maven:3.9.6-eclipse-temurin-21                                   2.5s
 => [internal] load metadata for docker.io/library/eclipse-temurin:21-jdk-jammy                                     2.7s
 => [internal] load .dockerignore                                                                                     0.1s
 => => transferring context: 2B                                                                                       0.0s
 => [build 1/5] FROM docker.io/library/maven:3.9.6-eclipse-temurin-21@sha256:8d63d4c1902cb12d9e79a70671b18ebe26358cb592561af33ca1808f00d935cb  0.3s
 => => resolve docker.io/library/maven:3.9.6-eclipse-temurin-21@sha256:8d63d4c1902cb12d9e79a70671b18ebe26358cb592561af33ca1808f00d935cb      0.3s
 => [internal] load build context                                                                                     0.1s
 => => transferring context: 1.49kB                                                                                   0.0s
 => [stage-1 1/3] FROM docker.io/library/eclipse-temurin:21-jdk-jammy@sha256:81ad1240d91eeafe1ab4154e9ed2310b67cb966caad1d235232ae10abcb1fae2  0.4s
 => => resolve docker.io/library/eclipse-temurin:21-jdk-jammy@sha256:81ad1240d91eeafe1ab4154e9ed2310b67cb966caad1d235232ae10abcb1fae2      0.3s
 => CACHED [stage-1 2/3] WORKDIR /app                                                                                 0.0s
 => CACHED [build 2/5] WORKDIR /app                                                                                   0.0s
 => CACHED [build 3/5] COPY pom.xml .                                                                                 0.0s
 => CACHED [build 4/5] COPY src ./src                                                                                 0.0s
 => [build 5/5] RUN mvn clean package -DskipTests                                                                   205.9s
 => [stage-1 3/3] COPY --from=build /app/target/*.jar app.jar                                                         9.2s
 => exporting to image                                                                                               23.7s
 => => exporting layers                                                                                              21.1s
 => => exporting manifest sha256:a26fa32d0c7af4207ff1aefbef02d074ca2a59b89ea13eb755769dc12471ab93                     0.5s
 => => exporting config sha256:aac2523ac787ed61c458c3c896726144ee54765c4acb07b026d426b04faf61ba                       0.1s
 => => exporting attestation manifest sha256:4b1fa7b50b2a49214b5554345a7c551b511f7db3d22f8bd026c3d0f09a964829         0.3s
 => => exporting manifest list sha256:f51aaf98b7af03638eed7bc450f26af294e92ac325149323dc0fae9996830ba7                0.2s
 => => naming to us-central1-docker.pkg.dev/devesh-192b2/movie-repo/movie-chat:latest                                0.0s
```

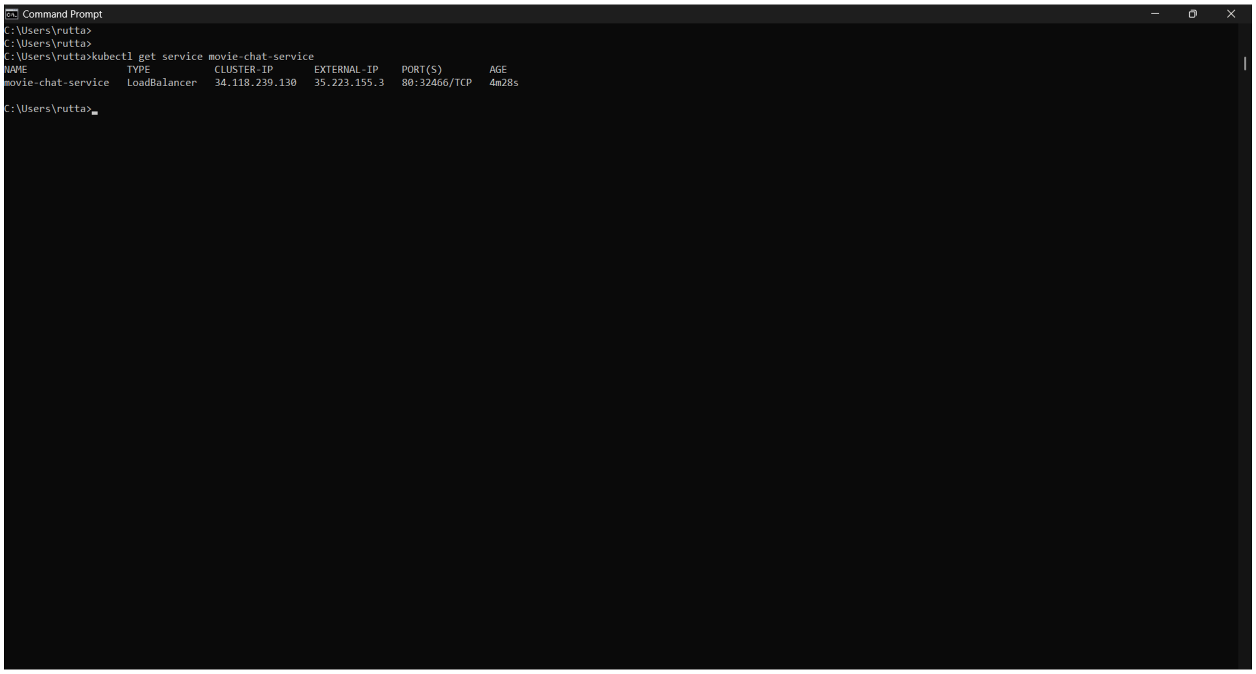


```yaml
version: '3.8'
services:
  postgres-db:
    image: postgres:16
    container_name: movie_chat_db
    environment:
      - POSTGRES_DB=moviedb
      - POSTGRES_USER=root
      - POSTGRES_PASSWORD=
    ports:
      - "5432:5432"
```

```
PS D:\devesh\moviechat> python agent_tool.py
 => => exporting attestation manifest sha256:4b1fa7b50b2a49214b5554345a7c551b511f7db3d22f8bd026c3d0f09a964829      0.3s
 => => exporting manifest list sha256:f51aaf98b7af03638eed7bc450f26af294e92ac325149323dc0fae9996830ba7             0.2s
 => => naming to us-central1-docker.pkg.dev/devesh-192b2/movie-repo/movie-chat:latest                             0.0s
 => => unpacking to us-central1-docker.pkg.dev/devesh-192b2/movie-repo/movie-chat:latest                          0.9s
✅ Success!

🚀 Authenticating Docker with GKE...
Adding credentials for: us-central1-docker.pkg.dev
Docker configuration file updated.
✅ Success!

🚀 Pushing Image to Google Artifact Registry...
The push refers to repository [us-central1-docker.pkg.dev/devesh-192b2/movie-repo/movie-chat]
02d80376fd82: Pushed
462f51c3d063: Pushed
01facf544478: Pushed
bac11bc655d8: Pushed
0974368270e4: Pushed
73435f0c25a7: Pushed
a55ed4a9da19: Pushed
7e49dc6156b0: Pushed
latest: digest: sha256:f51aaf98b7af03638eed7bc450f26af294e92ac325149323dc0fae9996830ba7 size: 856
✅ Success!
```

(Terminal running `agent_tool.py`) **Description:** *Execution of the automated Python deployment script (`agent_tool.py`). The logs show the script building the Docker image, pushing it to the Google Artifact Registry, and applying the Kubernetes manifests to the cluster.*

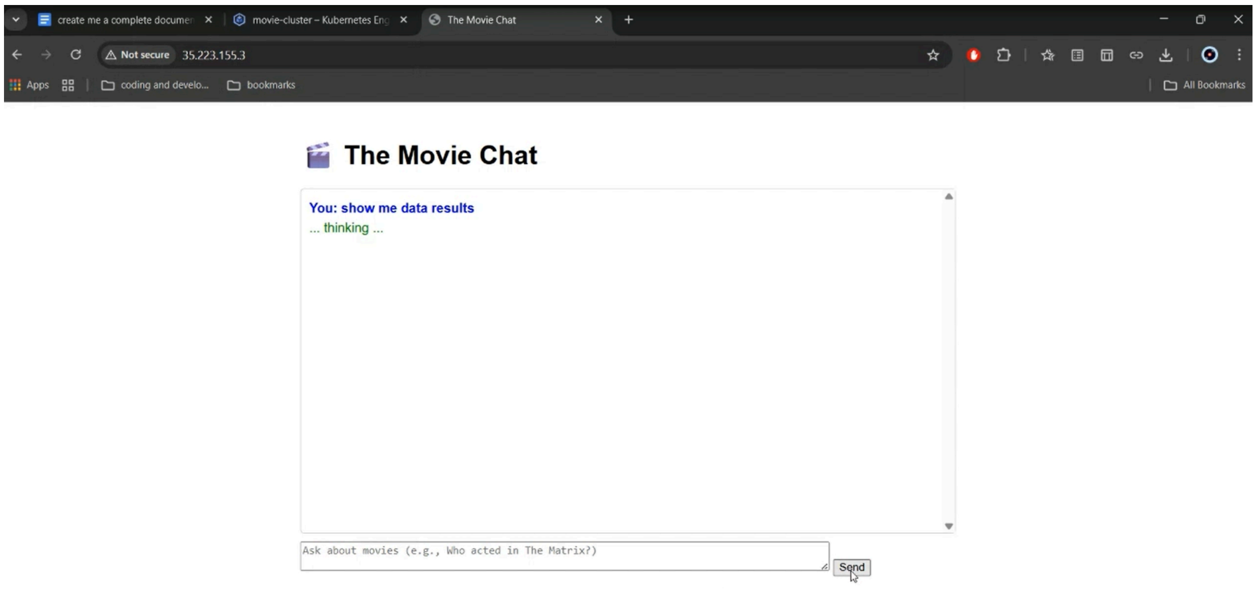
4. Live Cloud Deployment



```
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>kubectl get service movie-chat-service
NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
movie-chat-service   LoadBalancer   34.118.239.130   35.223.155.3   80:32466/TCP   4m28s

C:\Users\rutta>
```

(Terminal showing `kubectl get service`)

Description: *Terminal output verifying the successful deployment. The `kubectl get service` command shows the `movie-chat-service` has been assigned an external LoadBalancer IP (`34.118.239.130`), making it accessible from the internet.*


create me a complete documen
movie-cluster – Kubernetes Eng
The Movie Chat
Not secure
35.223.155.3
Apps
coding and develo...
bookmarks
All Bookmarks
Ask about movies (e.g., Who acted in The Matrix?)
Send

The Movie Chat

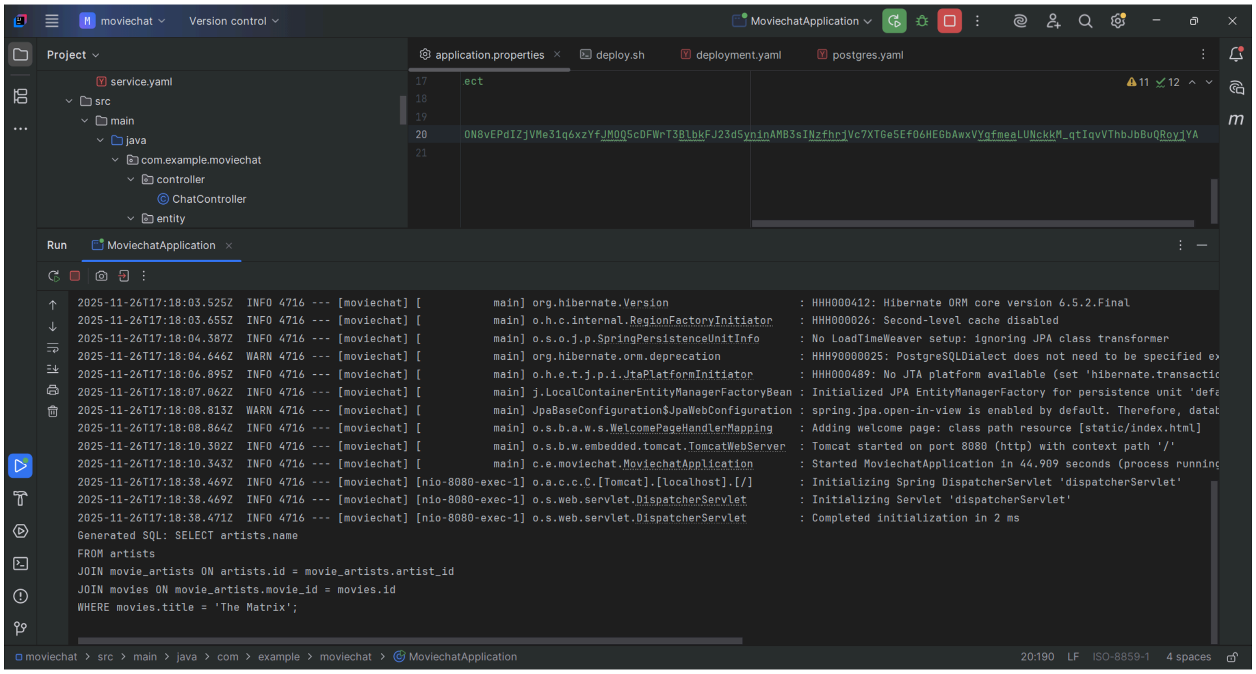
You: show me data results

... thinking ...

The Final Cloud Browser View: A screenshot of your browser address bar showing the external IP (e.g., `http://`(*`34.118.239.130`*)) with the app running. This is the most impactful "it works" image.

Demo Link to video Showcase [Here]()

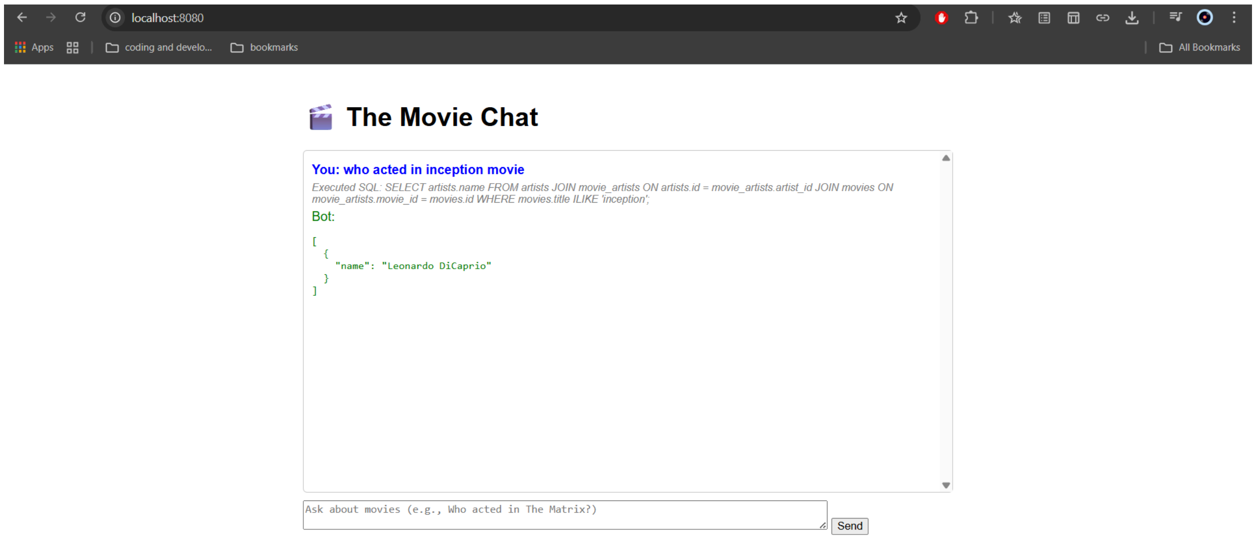
5. Backend Logs & SQL Generation and Clean UP



```
ect

ON8vEPdIZjVMe31q6xzYfJMOQ5cDFWrT3BlbkFJ23d5yninAMB3sINzfhrjVc7XTGe5Ef06HEGbAwxVYgfmeaLUNckkM_qtIqvVThbJbBuQRoyjYA
```

```
2025-11-26T17:18:03.525Z  INFO 4716 --- [moviechat] [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.5.2.Final
2025-11-26T17:18:03.655Z  INFO 4716 --- [moviechat] [           main] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
2025-11-26T17:18:04.387Z  INFO 4716 --- [moviechat] [           main] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-11-26T17:18:04.646Z  WARN 4716 --- [moviechat] [           main] org.hibernate.orm.deprecation            : HHH90000025: PostgreSQLDialect does not need to be specified ex
2025-11-26T17:18:06.895Z  INFO 4716 --- [moviechat] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transactic
2025-11-26T17:18:07.062Z  INFO 4716 --- [moviechat] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'defa
2025-11-26T17:18:08.813Z  WARN 4716 --- [moviechat] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, datab
2025-11-26T17:18:08.864Z  INFO 4716 --- [moviechat] [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/index.html]
2025-11-26T17:18:10.302Z  INFO 4716 --- [moviechat] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2025-11-26T17:18:10.343Z  INFO 4716 --- [moviechat] [           main] c.e.moviechat.MoviechatApplication       : Started MoviechatApplication in 44.909 seconds (process running
2025-11-26T17:18:38.469Z  INFO 4716 --- [moviechat] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-11-26T17:18:38.469Z  INFO 4716 --- [moviechat] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-11-26T17:18:38.471Z  INFO 4716 --- [moviechat] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
Generated SQL: SELECT artists.name
FROM artists
JOIN movie_artists ON artists.id = movie_artists.artist_id
JOIN movies ON movie_artists.movie_id = movies.id
WHERE movies.title = 'The Matrix';
```

(IntelliJ Logs showing "The Matrix" query) **Description:** *Backend logs capturing the AI-driven SQL generation. The logs show the exact SQL query (*`SELECT artists.name FROM artists ... WHERE movies.title = 'The Matrix'`*) generated in response to a user's request.*



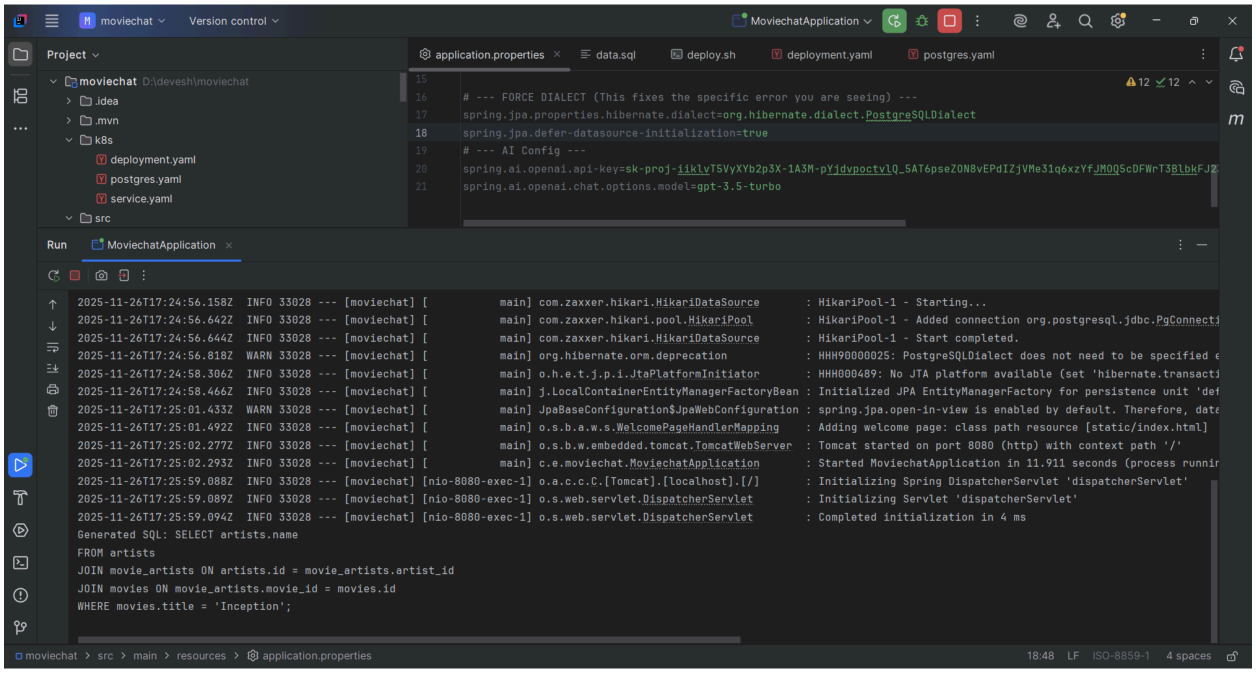
🎬 The Movie Chat

You: who acted in inception movie

Executed SQL: SELECT artists.name FROM artists JOIN movie_artists ON artists.id = movie_artists.artist_id JOIN movies ON movie_artists.movie_id = movies.id WHERE movies.title ILIKE 'inception';

Bot:

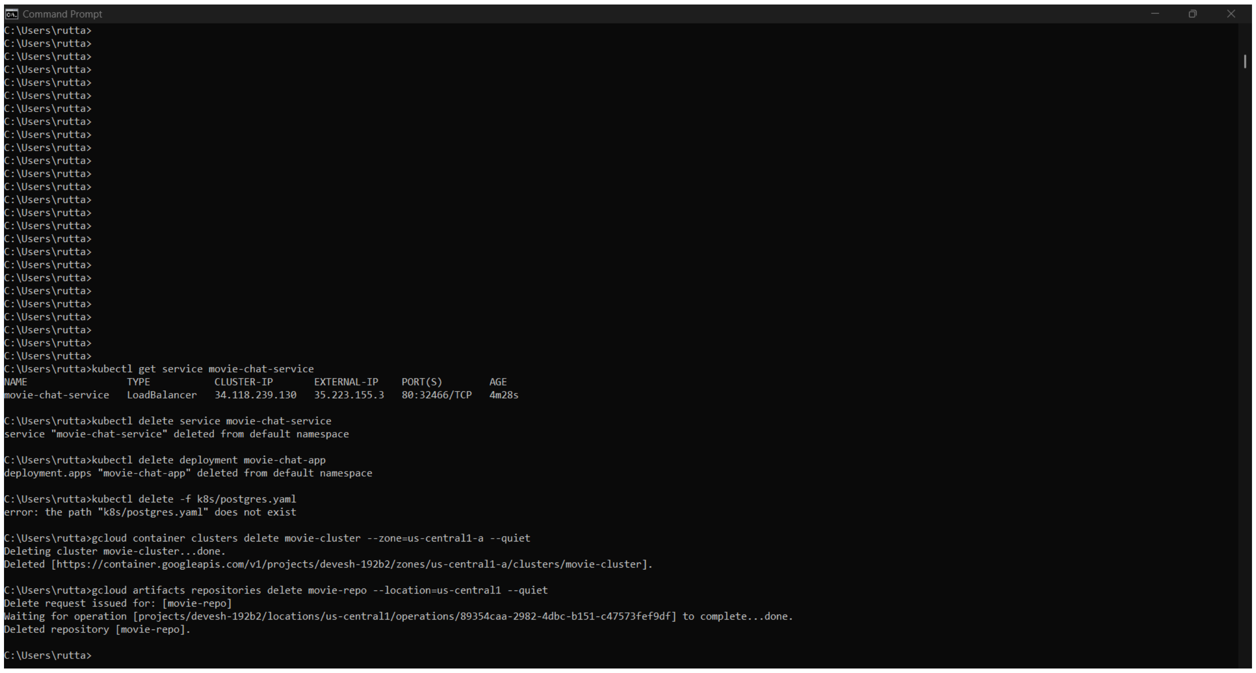
```
[
  {
    "name": "Leonardo DiCaprio"
  }
]
```



```
# --- FORCE DIALECT (This fixes the specific error you are seeing) ---
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.defer-datasource-initialization=true
# --- AI Config ---
spring.ai.openai.api-key=sk-proj-iiklvT5VyXYb2p3X-1A3M-pYjdvpoctvlQ_5AT6pseZON8vEPdIZjVMe31q6xzYfJMOQ5cDFWrT3BlbkFJ2
spring.ai.openai.chat.options.model=gpt-3.5-turbo
```

```
2025-11-26T17:24:56.158Z  INFO 33028 --- [moviechat] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-11-26T17:24:56.642Z  INFO 33028 --- [moviechat] [           main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection org.postgresql.jdbc.PgConnecti
2025-11-26T17:24:56.644Z  INFO 33028 --- [moviechat] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-11-26T17:24:56.818Z  WARN 33028 --- [moviechat] [           main] org.hibernate.orm.deprecation            : HHH90000025: PostgreSQLDialect does not need to be specified e
2025-11-26T17:24:58.306Z  INFO 33028 --- [moviechat] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transacti
2025-11-26T17:24:58.466Z  INFO 33028 --- [moviechat] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'def
2025-11-26T17:25:01.433Z  WARN 33028 --- [moviechat] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, data
2025-11-26T17:25:01.492Z  INFO 33028 --- [moviechat] [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/index.html]
2025-11-26T17:25:02.277Z  INFO 33028 --- [moviechat] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
2025-11-26T17:25:02.293Z  INFO 33028 --- [moviechat] [           main] c.e.moviechat.MoviechatApplication       : Started MoviechatApplication in 11.911 seconds (process runnin
2025-11-26T17:25:59.088Z  INFO 33028 --- [moviechat] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2025-11-26T17:25:59.089Z  INFO 33028 --- [moviechat] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-11-26T17:25:59.094Z  INFO 33028 --- [moviechat] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 4 ms
Generated SQL: SELECT artists.name
FROM artists
JOIN movie_artists ON artists.id = movie_artists.artist_id
JOIN movies ON movie_artists.movie_id = movies.id
WHERE movies.title = 'Inception';
```

(IntelliJ Logs showing "Inception" query) **Description:** *Another example of backend logs, showing the generated SQL for a question about the movie "Inception". This demonstrates the system's ability to handle various queries dynamically.*



```
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>
C:\Users\rutta>kubectl get service movie-chat-service
NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
movie-chat-service   LoadBalancer   34.118.239.130   35.223.155.3   80:32466/TCP   4m28s

C:\Users\rutta>kubectl delete service movie-chat-service
service "movie-chat-service" deleted from default namespace

C:\Users\rutta>kubectl delete deployment movie-chat-app
deployment.apps "movie-chat-app" deleted from default namespace

C:\Users\rutta>kubectl delete -f k8s/postgres.yaml
error: the path "k8s/postgres.yaml" does not exist

C:\Users\rutta>gcloud container clusters delete movie-cluster --zone=us-central1-a --quiet
Deleting cluster movie-cluster...done.
Deleted [https://container.googleapis.com/v1/projects/devesh-192b2/zones/us-central1-a/clusters/movie-cluster].

C:\Users\rutta>gcloud artifacts repositories delete movie-repo --location=us-central1 --quiet
Delete request issued for: [movie-repo]
Waiting for operation [projects/devesh-192b2/locations/us-central1/operations/89354caa-2982-4dbc-b151-c47573fef9df] to complete...done.
Deleted repository [movie-repo].

C:\Users\rutta>
```

Cleanup Verification: A final terminal screenshot showing the output of `gcloud container clusters list` as empty, proving you successfully deleted the resources to avoid cost