AI Meets node JS

Unlocking the Power of Large Language Models in JavaScript

POWERUp 2025

About Michael Dawson

Node.js lead for Red Hat and IBM

Active Node.js community member

- Node.js Collaborator, Node.js Technical Steering Committee,
- Active in a number of Working group(s)

Active OpenJS Foundation member

- Voting Cross Project Council Member
- Community Director 2020-2022

Twitter: @mhdawson1

GitHub: @mhdawson

Linkedin: https://www.linkedin.com/in/michael-dawson-6051282

Overview

- Why Node.js ?
- Our journey
    - Running a model locally
    - Leveraging a GPU
    - Some additional fundamentals, including Retrieval Augmented Generation
    - Working with different model serving options
    - Instrumenting the LLM components
    - A sample application - Parasol
    - A quick start with Podman AI Lab

Why Node.js ?

- Python often seen as runtime for AI
- But !!!
  - Not all applications will move to Python
  - Emerging AI client libraries often support TypeScript/JavaScript

What does it mean to me as a Node.js Dev

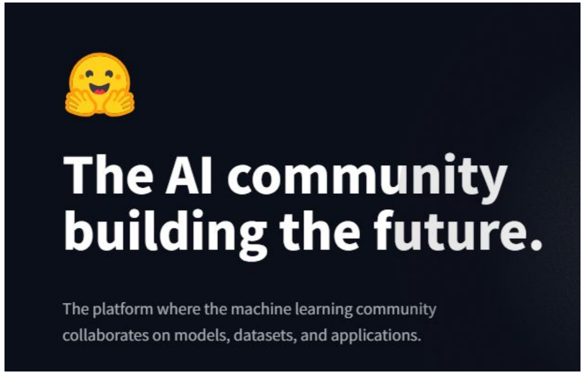
- In the beginning
  - https calls to bespoke service
- But now, many libraries are now available
  - langchain.com/LangGraph LangChain LangGraph
  - llamaindex.ai LlamaIndex
  - ollama ollama-js
  - Bee Agent
  - Llama Stack llama-stack

Running your first Langchain.js application

- Need a model or remote API service
  - Need to be cautious with proprietary info
  - Choose to start by running locally

Where do I get a model?

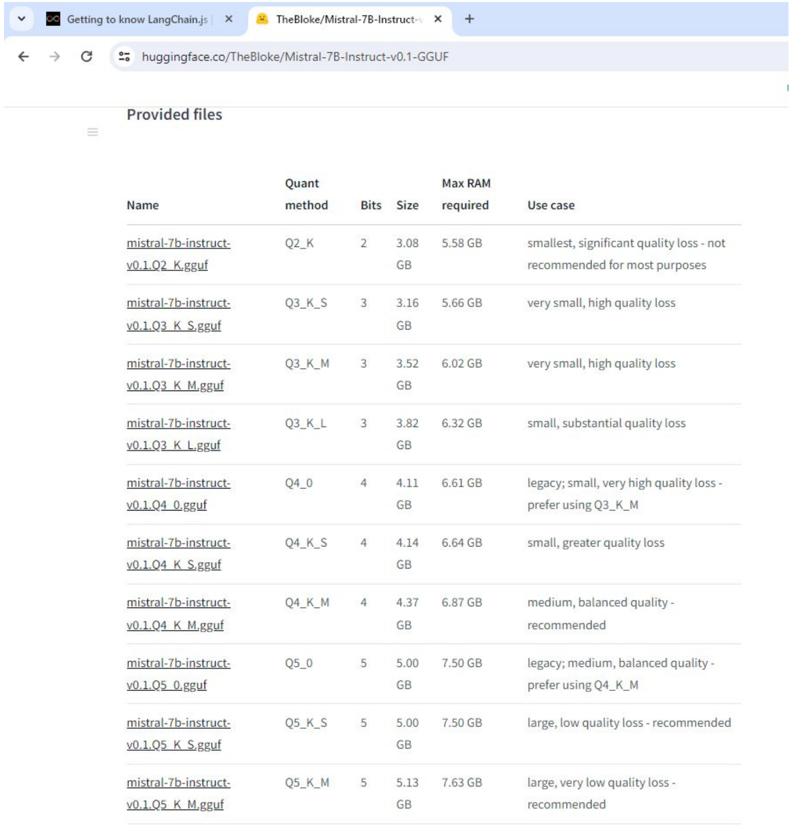
- HuggingFace



The AI community building the future.

The platform where the machine learning community collaborates on models, datasets, and applications.

Be careful of which model you use!


Getting to know LangChain.js
TheBloke/Mistral-7B-Instruct-
huggingface.co/TheBloke/Mistral-7B-Instruct-v0.1-GGUF

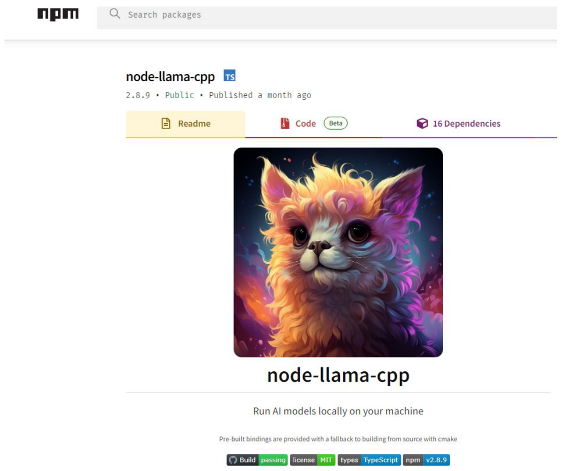
Provided files

Name	Quant method	Bits	Size	Max RAM required	Use case
mistral-7b-instruct-v0.1.Q2_K.gguf	Q2_K	2	3.08 GB	5.58 GB	smallest, significant quality loss - not recommended for most purposes
mistral-7b-instruct-v0.1.Q3_K_S.gguf	Q3_K_S	3	3.16 GB	5.66 GB	very small, high quality loss
mistral-7b-instruct-v0.1.Q3_K_M.gguf	Q3_K_M	3	3.52 GB	6.02 GB	very small, high quality loss
mistral-7b-instruct-v0.1.Q3_K_L.gguf	Q3_K_L	3	3.82 GB	6.32 GB	small, substantial quality loss
mistral-7b-instruct-v0.1.Q4_0.gguf	Q4_0	4	4.11 GB	6.61 GB	legacy; small, very high quality loss - prefer using Q3_K_M
mistral-7b-instruct-v0.1.Q4_K_S.gguf	Q4_K_S	4	4.14 GB	6.64 GB	small, greater quality loss
mistral-7b-instruct-v0.1.Q4_K_M.gguf	Q4_K_M	4	4.37 GB	6.87 GB	medium, balanced quality - recommended
mistral-7b-instruct-v0.1.Q5_0.gguf	Q5_0	5	5.00 GB	7.50 GB	legacy; medium, balanced quality - prefer using Q4_K_M
mistral-7b-instruct-v0.1.Q5_K_S.gguf	Q5_K_S	5	5.00 GB	7.50 GB	large, low quality loss - recommended
mistral-7b-instruct-v0.1.Q5_K_M.gguf	Q5_K_M	5	5.13 GB	7.63 GB	large, very low quality loss - recommended

How do I load a model?

- llama.ccp
- ollama
- Hugging Face transformers
- ...

- node-llama-cpp



node-llama-cpp

2.8.9 • Public • Published a month ago

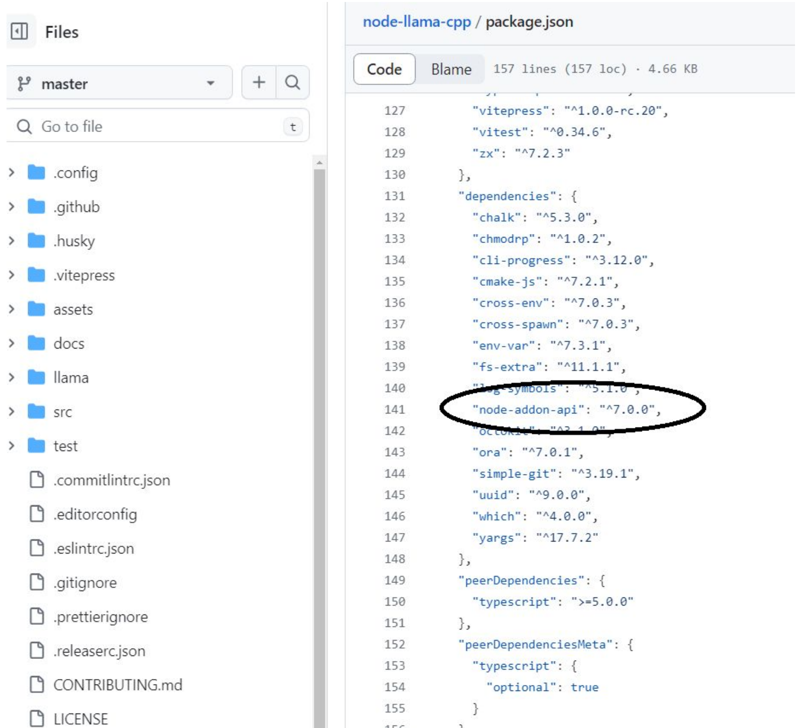
node-llama-cpp

Run AI models locally on your machine

Pre-built bindings are provided with a fallback to building from source with cmake

What’s this under the covers ?

- node-addon-api



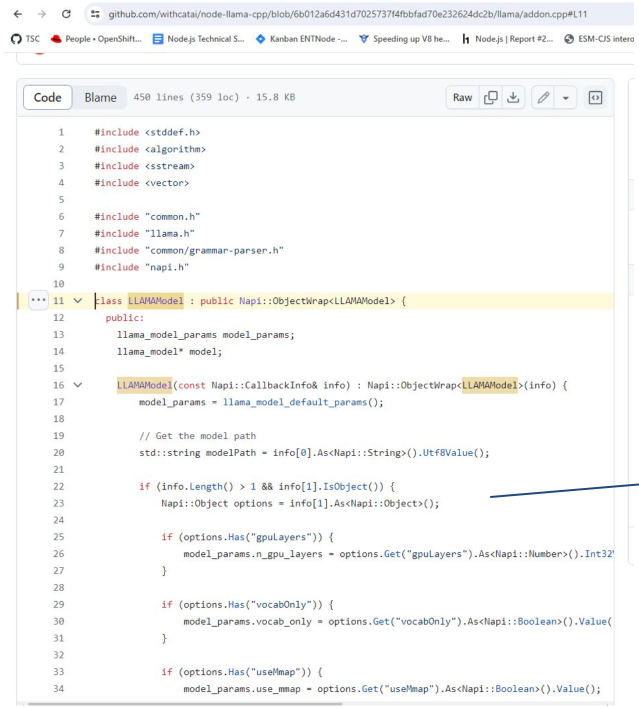
```
    "vitepress": "^1.0.0-rc.20",
    "vitest": "^0.34.6",
    "zx": "^7.2.3"
  },
  "dependencies": {
    "chalk": "^5.3.0",
    "chmodrp": "^1.0.2",
    "cli-progress": "^3.12.0",
    "cmake-js": "^7.2.1",
    "cross-env": "^7.0.3",
    "cross-spawn": "^7.0.3",
    "env-var": "^7.3.1",
    "fs-extra": "^11.1.1",
    "log-symbols": "^5.1.0",
    "node-addon-api": "^7.0.0",
    "octokit": "^3.1.0",
    "ora": "^7.0.1",
    "simple-git": "^3.19.1",
    "uuid": "^9.0.0",
    "which": "^4.0.0",
    "yargs": "^17.7.2"
  },
  "peerDependencies": {
    "typescript": ">=5.0.0"
  },
  "peerDependenciesMeta": {
    "typescript": {
      "optional": true
    }
```

The state of the Node.js core: The Monthly Dev #29

[Building Native addons like its 2023]()

node-addon-api - https://github.com/nodejs/node-addon-api

What’s this under the covers ?



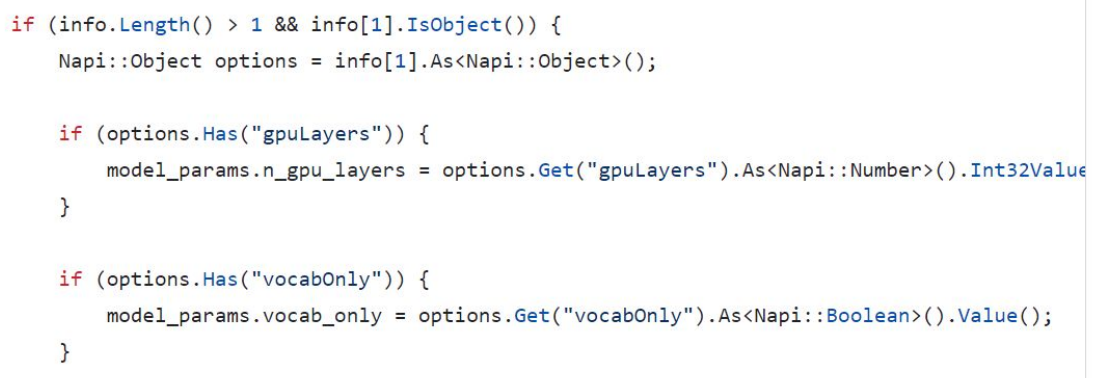
github.com/withcatai/node-llama-cpp/blob/6b012a6d431d7025737f4fbbfad70e232624dc2b/llama/addon.cpp#L11

TSC · People • OpenShift... · Node.js Technical S... · Kanban ENTNode -... · Speeding up V8 he... · Node.js | Report #2... · ESM-CJS intero

Code · Blame · 450 lines (359 loc) · 15.8 KB · Raw

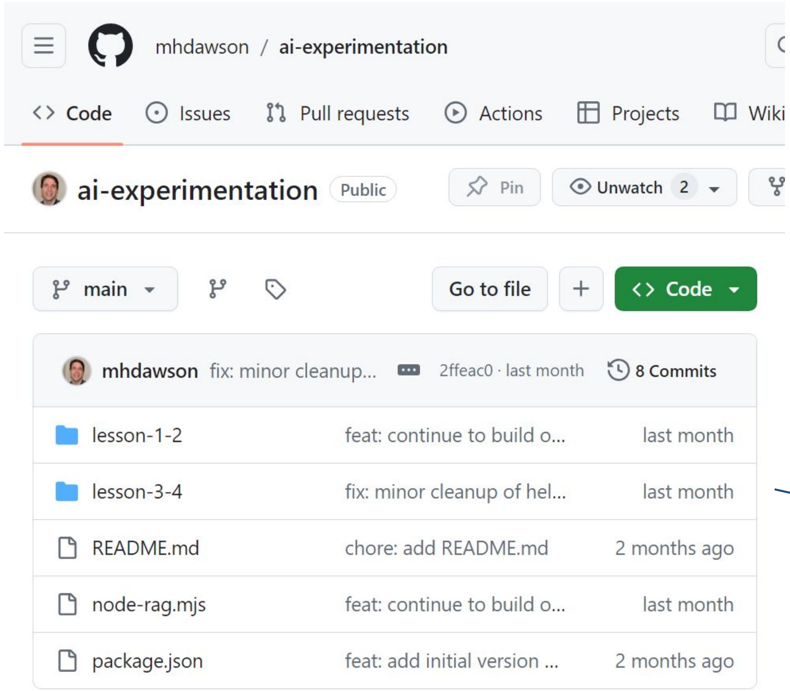
```cpp
#include <stddef.h>
#include <algorithm>
#include <sstream>
#include <vector>

#include "common.h"
#include "llama.h"
#include "common/grammar-parser.h"
#include "napi.h"

class LLAMAModel : public Napi::ObjectWrap<LLAMAModel> {
  public:
    llama_model_params model_params;
    llama_model* model;

    LLAMAModel(const Napi::CallbackInfo& info) : Napi::ObjectWrap<LLAMAModel>(info) {
        model_params = llama_model_default_params();

        // Get the model path
        std::string modelPath = info[0].As<Napi::String>().Utf8Value();

        if (info.Length() > 1 && info[1].IsObject()) {
            Napi::Object options = info[1].As<Napi::Object>();

            if (options.Has("gpuLayers")) {
                model_params.n_gpu_layers = options.Get("gpuLayers").As<Napi::Number>().Int32
            }

            if (options.Has("vocabOnly")) {
                model_params.vocab_only = options.Get("vocabOnly").As<Napi::Boolean>().Value(
            }

            if (options.Has("useMmap")) {
                model_params.use_mmap = options.Get("useMmap").As<Napi::Boolean>().Value();
```

https://github.com/withcatai/node-llama-cpp/blob/6b012a6d431d7025737f4fbbfad70e232624dc2b/llama/addon.cpp



```cpp
if (info.Length() > 1 && info[1].IsObject()) {
    Napi::Object options = info[1].As<Napi::Object>();

    if (options.Has("gpuLayers")) {
        model_params.n_gpu_layers = options.Get("gpuLayers").As<Napi::Number>().Int32Value
    }

    if (options.Has("vocabOnly")) {
        model_params.vocab_only = options.Get("vocabOnly").As<Napi::Boolean>().Value();
    }
```

Where's the code ?



mhdawson / ai-experimentation

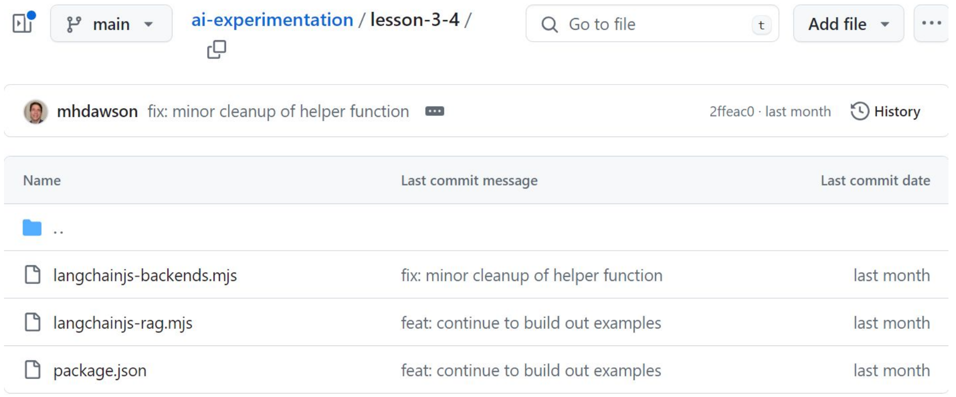
Code | Issues | Pull requests | Actions | Projects | Wiki

ai-experimentation Public

main | Go to file | + | Code

mhdawson fix: minor cleanup... 2ffeac0 · last month 8 Commits

lesson-1-2	feat: continue to build o...	last month
lesson-3-4	fix: minor cleanup of hel...	last month
README.md	chore: add README.md	2 months ago
node-rag.mjs	feat: continue to build o...	last month
package.json	feat: add initial version ...	2 months ago



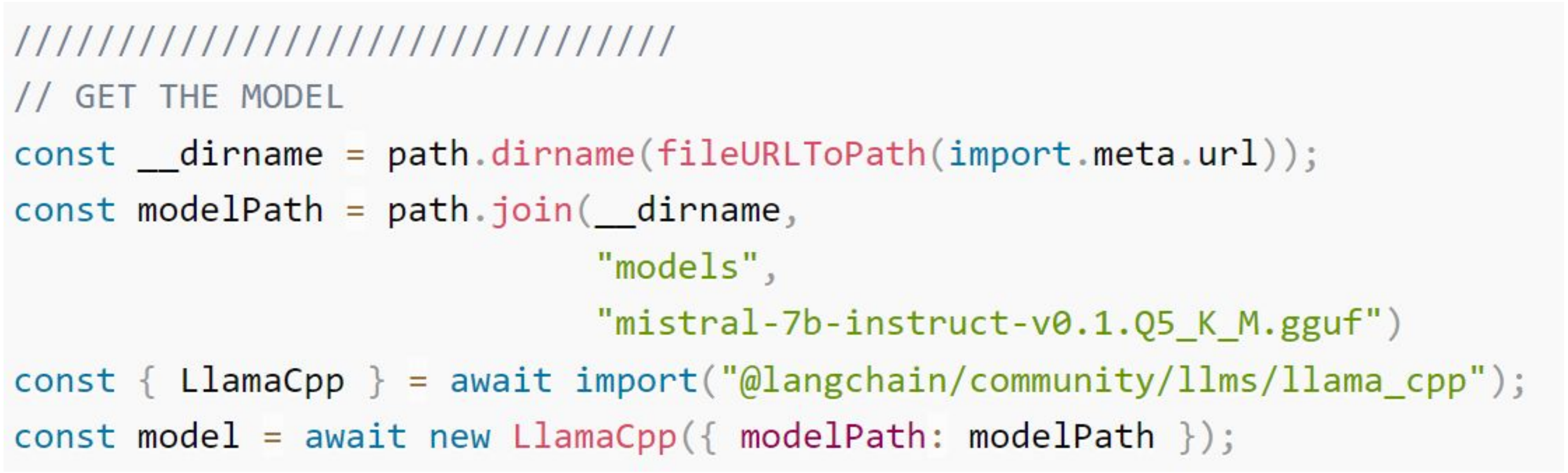
main | ai-experimentation / lesson-3-4 / | Go to file | Add file

mhdawson fix: minor cleanup of helper function 2ffeac0 · last month History

Name	Last commit message	Last commit date
..		
langchainjs-backends.mjs	fix: minor cleanup of helper function	last month
langchainjs-rag.mjs	feat: continue to build out examples	last month
package.json	feat: continue to build out examples	last month

https://github.com/mhdawson/ai-experimentation

Loading the model in Langchain.js

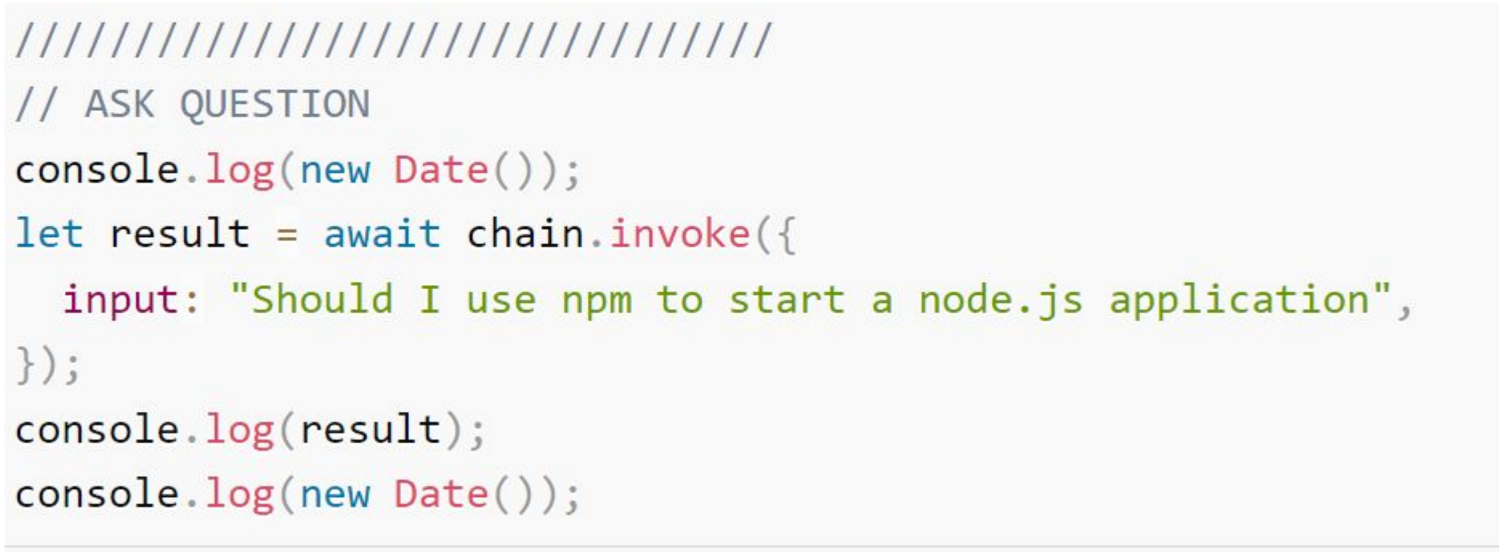


```
////////////////////////////////
// GET THE MODEL
const __dirname = path.dirname(fileURLToPath(import.meta.url));
const modelPath = path.join(__dirname,
                            "models",
                            "mistral-7b-instruct-v0.1.Q5_K_M.gguf")
const { LlamaCpp } = await import("@langchain/community/llms/llama_cpp");
const model = await new LlamaCpp({ modelPath: modelPath });
```

Asking my first question - create a chain

```
/////////////////////////////////
// CREATE CHAIN
const prompt =
  ChatPromptTemplate.fromTemplate(`Answer the following
question if you don't know the answer say so:
Question: {input}`);
const chain = prompt.pipe(model);
```

Asking my first question - ask a question



```javascript
////////////////////////////////
// ASK QUESTION
console.log(new Date());
let result = await chain.invoke({
  input: "Should I use npm to start a node.js application",
});
console.log(result);
console.log(new Date());
```

Asking my first question - 25 seconds later….

```
2024-03-11T22:08:23.372Z
Assistant: Yes, you should use npm to start a Node.js application. NPM (Node Package Manager) is the
default package manager for Node.js and it provides a centralized repository of packages that can be
used in your applications. It also allows you to manage dependencies between packages and automate
tasks such as testing and deployment. If you are new to Node.js, I would recommend using npm to get
started with your application development.
2024-03-11T22:08:45.774Z
```

Not the answer we want people to get based on the

nodejs-reference-architecture

https://github.com/nodeshift/nodejs-reference-architecture

JSDrops Growing Success Across Organizations -
https://www.youtube.com/watch?v=GncwXJBwcgQ

Hmm, 25 seconds is a bit long

- Good news, node-llama-cpp supports GPUs
  - enabled by default for MacOS (non intel)
  - easy to enable for Windows

Turning on the GPU windows/NVIDIA

- install the CUDA toolkit (version 12.x or higher).
- install the C/C++ compiler for your platform, including support for CMake and CMake.js.
- npx --no node-llama-cpp download --cuda

Turning on the GPU

25 → 3 Seconds

NVIDIA 4060Ti 16G

Fast but still wrong answer!

- Often want to add additional knowledge
    - Building/Training a model is a lot of work
    - Just want to add a bit of specific info

Some Fundamentals - Overview

- Additional Context
- Message History
- Function Calling

Retrieval Augmented Generation (RAG)

- Indexing
  - Load and Store Documents
  - Happens “Offline”
- Retrieval and Generation
  - Extract relevant chunks
  - Add chunks to prompt context

Note! - supported context is limited, for example 2k

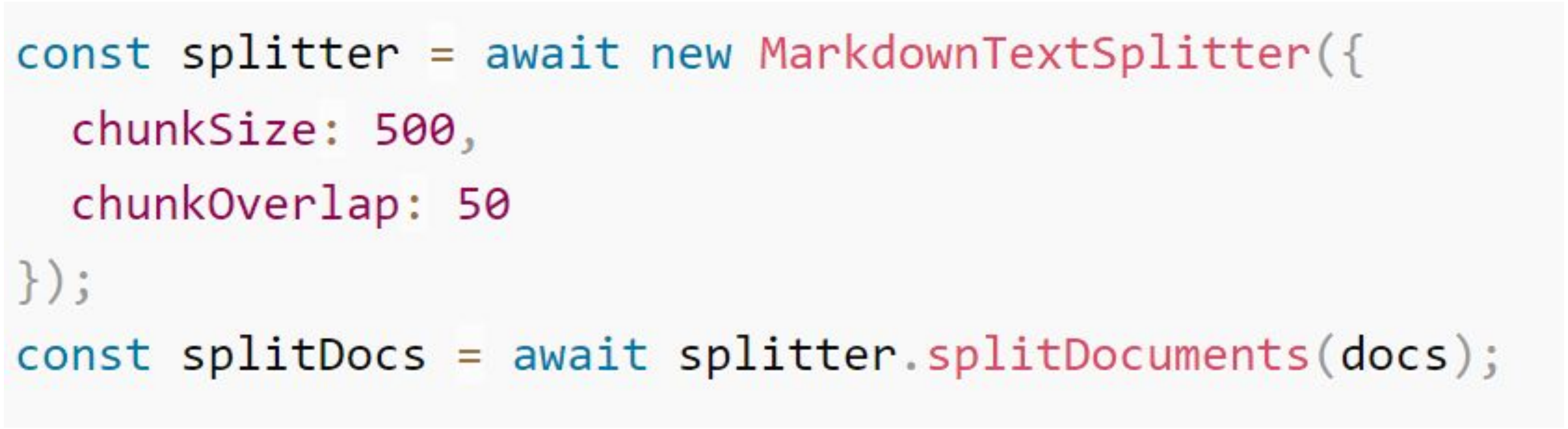
https://js.langchain.com/docs/tutorials/rag/

Load the documents

nodejs-reference-architecture

```
const docLoader = new DirectoryLoader(
  "./SOURCE_DOCUMENTS",
  {
    ".md": (path) => new TextLoader(path),
  }
);
const docs = await docLoader.load();
```

Split the documents



```
const splitter = await new MarkdownTextSplitter({
  chunkSize: 500,
  chunkOverlap: 50
});
const splitDocs = await splitter.splitDocuments(docs);
```

Store the chunks in a database

```
const vectorStore = await MemoryVectorStore.fromDocuments(
  splitDocs,
  new HuggingFaceTransformersEmbeddings()
);
```

Create Chain and Retrieve Docs

```
const prompt =
  ChatPromptTemplate.fromTemplate(`Answer the following
question based only on the provided context, if you
don't know the answer say so:

<context>
{context}
</context>

Question: {input}`);

const documentChain = await createStuffDocumentsChain({
  llm: model,
  prompt,
});
```

```
const retriever = await vectorStore.asRetriever();



const retrievalChain = await createRetrievalChain({
  combineDocsChain: documentChain,
  retriever,
});
```

Ask Question

```
////////////////////////////////
// ASK QUESTIONS

console.log(new Date());
let result = await retrievalChain.invoke({
  input: "Should I use npm to start a node.js application",
});
console.log(result);
console.log(new Date());
```

A Better Answer

'Assistant: It is generally not necessary to use `npm` to start a Node.js application. If you avoid using it in the container, you will not be exposed to any security vulnerabilities that might exist in that component or its dependencies. However, it is important to build security into your software development process when developing Node.js modules and applications. This includes managing dependencies, managing access and content of public and private data stores such as npm and github, writing defensive code, limiting required execution privileges, supporting logging and monitoring, and externalizing secrets.'

Message History

- LLMs are Stateless
  - No context of previous questions

Example:

- Human: My name is Luke
- AI: Hello, Luke.
- Human: What’s my name?
- AI: Huh? You never told me.

Message History

```
const prompt = ChatPromptTemplate.fromMessages([
  [ 'system', 'You are an assistant….],
  new MessagesPlaceholder('history'),
  [ 'human', '{input}' ]
]);

const sessions = {};



const chainWithHistory = new RunnableWithMessageHistory({
  runnable: chain,
  getMessageHistory: (sessionId) => {...},
  inputMessagesKey: 'input',
  historyMessagesKey: 'history',
});
```

```
const response1 = await chainWithHistory.invoke(
  { input: 'My name is Luke' },
  {
    configurable: {
      sessionId: 'funtimes'
    }
  }
);

console.log('Response 1: ', response1);
```

```
const response2 = await chainWithHistory.invoke(
  { input: 'What is my name' },
  {
    configurable: {
      sessionId: 'funtimes'
    }
  }
);

console.log('Response 2: ', response2);

console.log(sessions);
```

https://github.com/lholmquist/ascend_ai_talk/blob/main/message_history/history.js

Message History

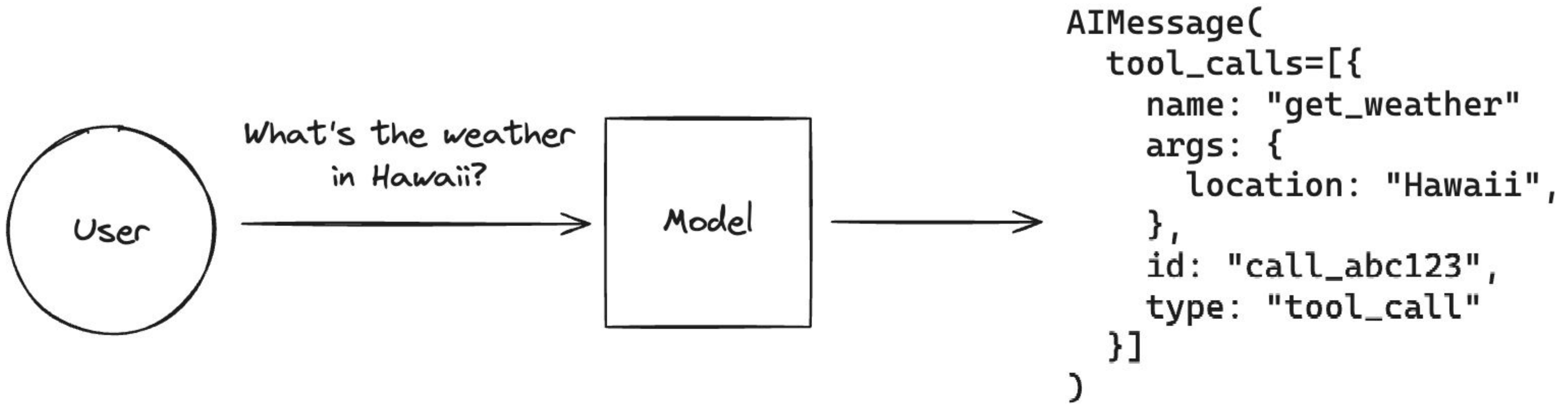
```
Response 1:   Hello, Luke! How can I assist
you today?


Response 2:   Your name is Luke. Is there
something specific you would like help with?
```

```
{
  funtimes: InMemoryChatMessageHistory {
    messages: [
      HumanMessage {
        "content": "My name is Luke",
      },
      AIMessage {
        "content": " Hello, Luke! How can I assist you
today?",
      },
      HumanMessage {
        "content": "What is my name",
      },
      AIMessage {
        "content": " Your name is Luke. Is there something
specific you would like help with?"
      }
    ]
  }
}
```

https://github.com/lholmquist/ascend_ai_talk/blob/main/message_history/history.js

Agents and Function Calling

- Respond by "calling a tool"
- LLM doesn't run the tool
  - generates the arguments
- Not Universal
  - Supported by many popular LLM providers



```
AIMessage(
  tool_calls=[{
    name: "get_weather"
    args: {
      location: "Hawaii",
    },
    id: "call_abc123",
    type: "tool_call"
  }]
)
```

https://js.langchain.com/docs/how_to/tool_calling/

Agents and Function Calling

```
const calculatorTool = tool(
  async ({ operation, number1, number2 }) => {
    if (operation === "add") {
      return `${number1 + number2}`;
    } else if (operation === "subtract") {
      return `${number1 - number2}`;
    } else if (operation === "multiply") {
      return `${number1 * number2}`;
    } else if (operation === "divide") {
      return `${number1 / number2}`;
    } else {
      throw new Error("Invalid operation.");
    }
  },
  {
    name: "calculator",
    description: "Can perform mathematical operations.",
    schema: calculatorSchema,
  }
);

const llmWithTools = model.bindTools([calculatorTool]);
```

https://github.com/lholmquist/ascend_ai_talk/blob/main/tools/tools.js

```
const messages = [new HumanMessage("What is 3 * 12?")];

const aiMessage = await llmWithTools.invoke(messages);

AIMessage {
  "content": "",
  "additional_kwargs": {
    "tool_calls": [
      { "id": "call_kfmpr9ux",
        "type": "function" },
    ]
  },
  "tool_calls": [
    {
      "name": "calculator",
      "args": {
        "number1": 3,
        "number2": 12,
        "operation": "multiply"
      },
      "type": "tool_call",
      "id": "call_kfmpr9ux"
    },
   ],
}
```

Agents and Function Calling

```
for (const toolCall of aiMessage.tool_calls) {
  const toolMessage = await calculatorTool.invoke(toolCall);

  console.log(toolMessage);

  messages.push(toolMessage);

}
```

```
Value of toolCall:
{
  name: 'calculator',
  args: {
    number1: 3,
    number2: 12,
    operation: 'multiply'
  },
  type: 'tool_call',
  id: 'call_kfmpr9ux'
}
```

```
  ToolMessage {
      "content": "36",
      "name": "calculator",
      "additional_kwargs": {},
      "response_metadata": {},
      "tool_call_id": "call_kfmpr9ux"
    },
```

Agents and Function Calling

```
[
  HumanMessage {
    "content": "What is 3 * 12?"
  },
  AIMessage {
    "id": "chatcmpl-352",
    "content": "",
    "additional_kwargs": {
      "tool_calls": [
        {
          "id": "call_kfmpr9ux",
          "type": "function"
        },
      ]
    },
    "tool_calls": [
      {
        "name": "calculator",
        "args": {
          "number1": 3,
          "number2": 12,
          "operation": "multiply"
        },
        "type": "tool_call",
        "id": "call_kfmpr9ux"
      },
    ],
  },
  ToolMessage {
    "content": "36",
    "name": "calculator",
    "tool_call_id": "call_kfmpr9ux"
  },
]
```

```
const result = await llmWithTools.invoke(messages);
console.log(result);
```

```
AIMessage {
  "id": "chatcmpl-497",
  "content": "36 is the result of multiplying 3 by 12"
  },
  "tool_calls": []
}
```

LangGraph

[LangGraph.js]()

Switching to other model serving back ends

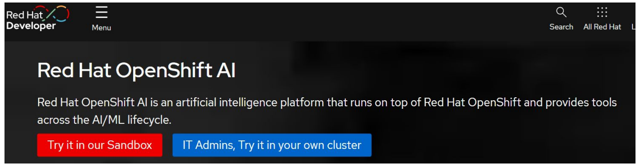
```
async function getModel(type, temperature) {
  console.log("Loading model - " + new Date());

  let model;
  if (type === 'llama-cpp') {
    const __dirname = path.dirname(fileURLToPath(import.meta.url));
    const modelPath = path.join(__dirname, "models", "mistral-7b-instruct-v0.1.Q5_K_M.gguf")
    const { LlamaCpp } = await import("@langchain/community/llms/llama_cpp");
    model = await new LlamaCpp({ modelPath: modelPath,
                                 batchSize: 1024,
                                 temperature: temperature,
                                 gpuLayers: 64 });
  } else if (type === 'openAI') {
    const { ChatOpenAI } = await import("@langchain/openai");
    const key = await import('../key.json', { with: { type: 'json' } });
    model = new ChatOpenAI({
      temperature: temperature,
      openAIApiKey: key.default.apiKey
    });
```

https://github.com/mhdawson/ai-experimentation/blob/main/lesson-5/langchainjs-ollama.mjs

Switching to other model serving back ends

```
 } else if (type === 'Openshift.ai') {
  ////////////////////////////////
  // Connect to OpenShift.ai endpoint
  const { ChatOpenAI } = await import("@langchain/openai");
  model = new ChatOpenAI(
    { temperature: temperature,
      openAIApiKey: 'EMPTY',
      modelName: 'mistralai/Mistral-7B-Instruct-v0.2' },
    { baseURL: 'http://vllm.llm-hosting.svc.cluster.local:8000/v1' }
  );
} else if (type === 'ollama') {
  ////////////////////////////////
  // Connect to ollama endpoint
  const { Ollama } = await import("@langchain/community/llms/ollama");
  model = new Ollama({
    baseUrl: "http://10.1.1.39:11434", // Default value
    model: "mistral", // Default value
  });
};
```

https://developers.redhat.com/products/red-hat-openshift-ai/overview

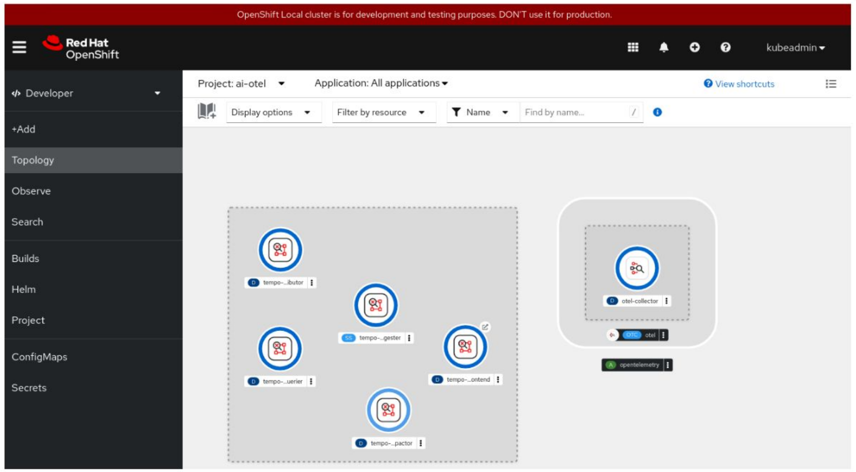


Red Hat OpenShift AI

Red Hat OpenShift AI is an artificial intelligence platform that runs on top of Red Hat OpenShift and provides tools across the AI/ML lifecycle.

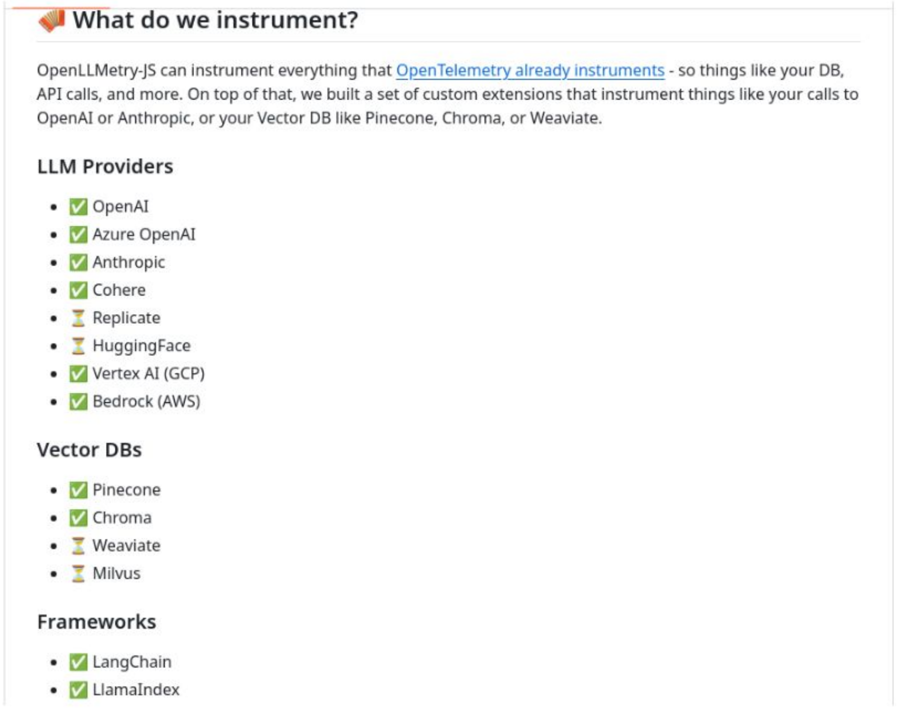
Instrumenting the LLM components

- OpenTelemetry
  - Trending to become common standard
- Red Hat build of OpenTelemetry


OpenShift Local cluster is for development and testing purposes. DON'T use it for production.

Instrumenting the LLM components

- LLM instrumentation following the trend
  - [openllmetry]
  - [langtrace]



What do we instrument?

OpenLLMetry-JS can instrument everything that OpenTelemetry already instruments - so things like your DB, API calls, and more. On top of that, we built a set of custom extensions that instrument things like your calls to OpenAI or Anthropic, or your Vector DB like Pinecone, Chroma, or Weaviate.

LLM Providers

- ✅ OpenAI
- ✅ Azure OpenAI
- ✅ Anthropic
- ✅ Cohere
- ⏳ Replicate
- ⏳ HuggingFace
- ✅ Vertex AI (GCP)
- ✅ Bedrock (AWS)

Vector DBs

- ✅ Pinecone
- ✅ Chroma
- ⏳ Weaviate
- ⏳ Milvus

Frameworks

- ✅ LangChain
- ✅ LlamaIndex

https://github.com/traceloop/openllmetry-js#-what-do-we-instrument

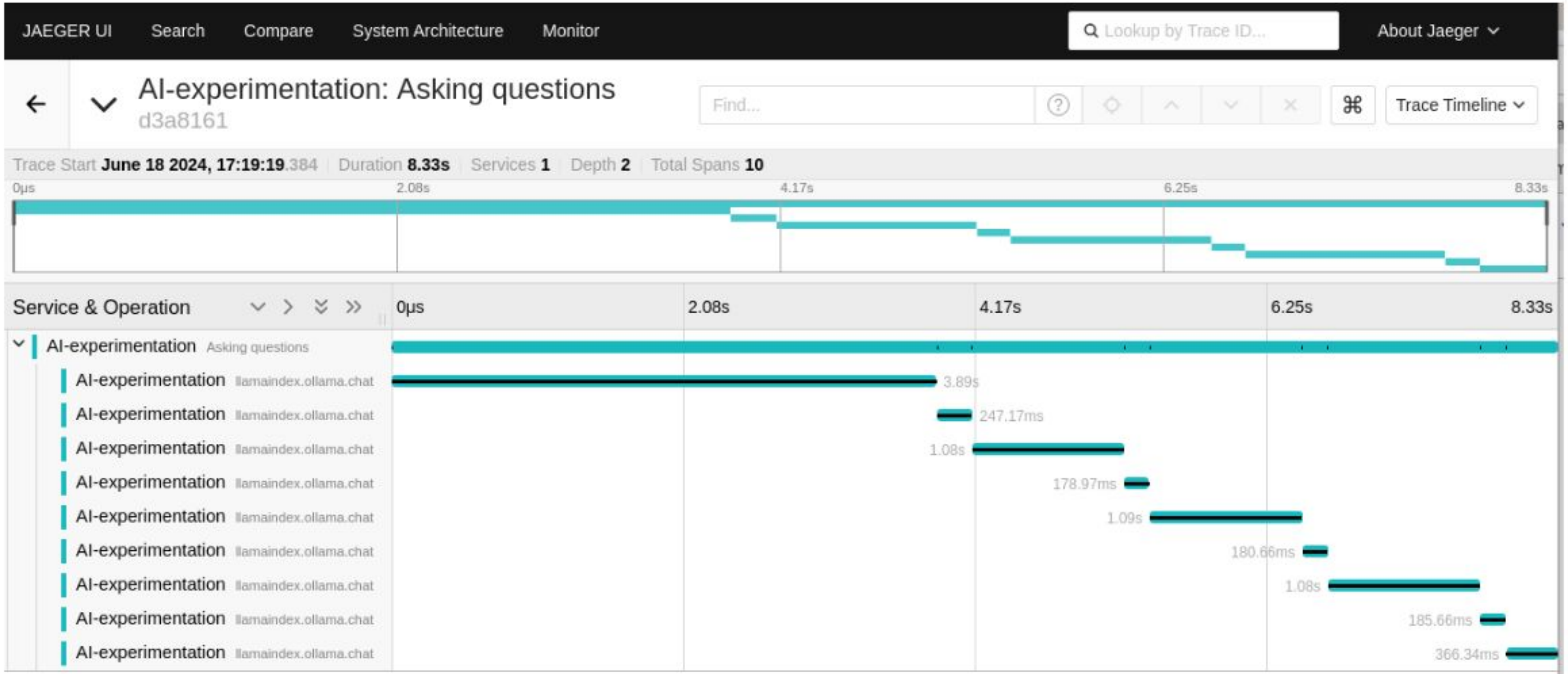
Instrumenting the LLM components

Auto Instrumentation

```
import * as traceloop from "@traceloop/node-server-sdk";
import { trace, context } from "@opentelemetry/api";
import * as LlamaIndex from "llamaindex";
import { OTLPTraceExporter } from "@opentelemetry/exporter-trace-otlp-grpc";
import { ConsoleSpanExporter } from "@opentelemetry/sdk-trace-node";

traceloop.initialize({ exporter: new OTLPTraceExporter(),
                        disableBatch: true,
                        instrumentModules: {
                          llamaIndex: LlamaIndex,
                        }
                      });

const tracer = trace.getTracer();

import {askQuestions} from "./llamaindex-function-ollama.mjs"
tracer.startActiveSpan('Asking questions', async (span) => {
  await askQuestions();
  span.end();
});
```

$\longleftarrow$ Often Optional

Instrumenting the LLM components

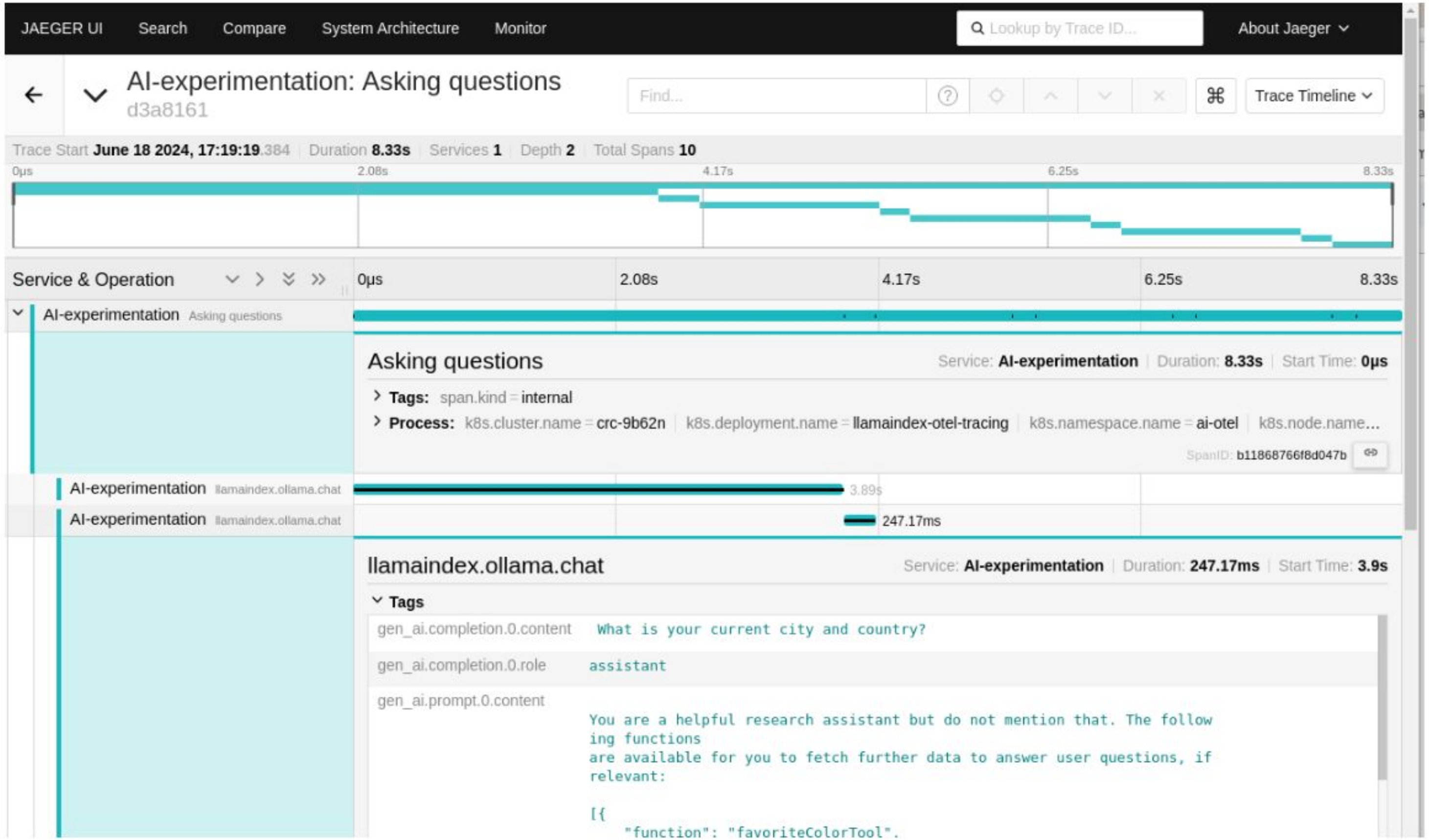


AI-experimentation: Asking questions
d3a8161

Trace Start **June 18 2024, 17:19:19**.384 | Duration **8.33s** | Services **1** | Depth **2** | Total Spans **10**

Service & Operation	Duration
AI-experimentation Asking questions	
AI-experimentation llamaindex.ollama.chat	3.89s
AI-experimentation llamaindex.ollama.chat	247.17ms
AI-experimentation llamaindex.ollama.chat	1.08s
AI-experimentation llamaindex.ollama.chat	178.97ms
AI-experimentation llamaindex.ollama.chat	1.09s
AI-experimentation llamaindex.ollama.chat	180.66ms
AI-experimentation llamaindex.ollama.chat	1.08s
AI-experimentation llamaindex.ollama.chat	185.66ms
AI-experimentation llamaindex.ollama.chat	366.34ms

Instrumenting the LLM components



AI-experimentation: Asking questions
d3a8161

Trace Start **June 18 2024, 17:19:19**.384 | Duration **8.33s** | Services **1** | Depth **2** | Total Spans **10**

Service & Operation | 0µs | 2.08s | 4.17s | 6.25s | 8.33s

AI-experimentation Asking questions

Asking questions — Service: **AI-experimentation** | Duration: **8.33s** | Start Time: **0µs**

Tags: span.kind = internal

Process: k8s.cluster.name = crc-9b62n | k8s.deployment.name = llamaindex-otel-tracing | k8s.namespace.name = ai-otel | k8s.node.name...

SpanID: b11868766f8d047b

AI-experimentation llamaindex.ollama.chat — 3.89s

AI-experimentation llamaindex.ollama.chat — 247.17ms

llamaindex.ollama.chat — Service: **AI-experimentation** | Duration: **247.17ms** | Start Time: **3.9s**

Tags

gen_ai.completion.0.content	What is your current city and country?
gen_ai.completion.0.role	assistant
gen_ai.prompt.0.content	

```
You are a helpful research assistant but do not mention that. The follow
ing functions
are available for you to fetch further data to answer user questions, if
relevant:

[{
    "function": "favoriteColorTool".
```

IBM i as Client

- favorite-color.mjs - https://github.com/mhdawson/ai-tool-experimentation/blob/main/ollama/favorite-color.mjs
- Tests function calling
  - favoriteColorTool
  - favoriteHockeyTeamTool
- Questions

```
'What is my favorite color?',
'My city is Ottawa',
'My country is Canada',
'I moved to Montreal. What is my favorite color now?',
'My city is Montreal and my country is Canada',
'My city is Ottawa and my country is Canada, what is my favorite color?',
'What is my favorite hockey team ?',
'My city is Montreal and my country is Canada',
```

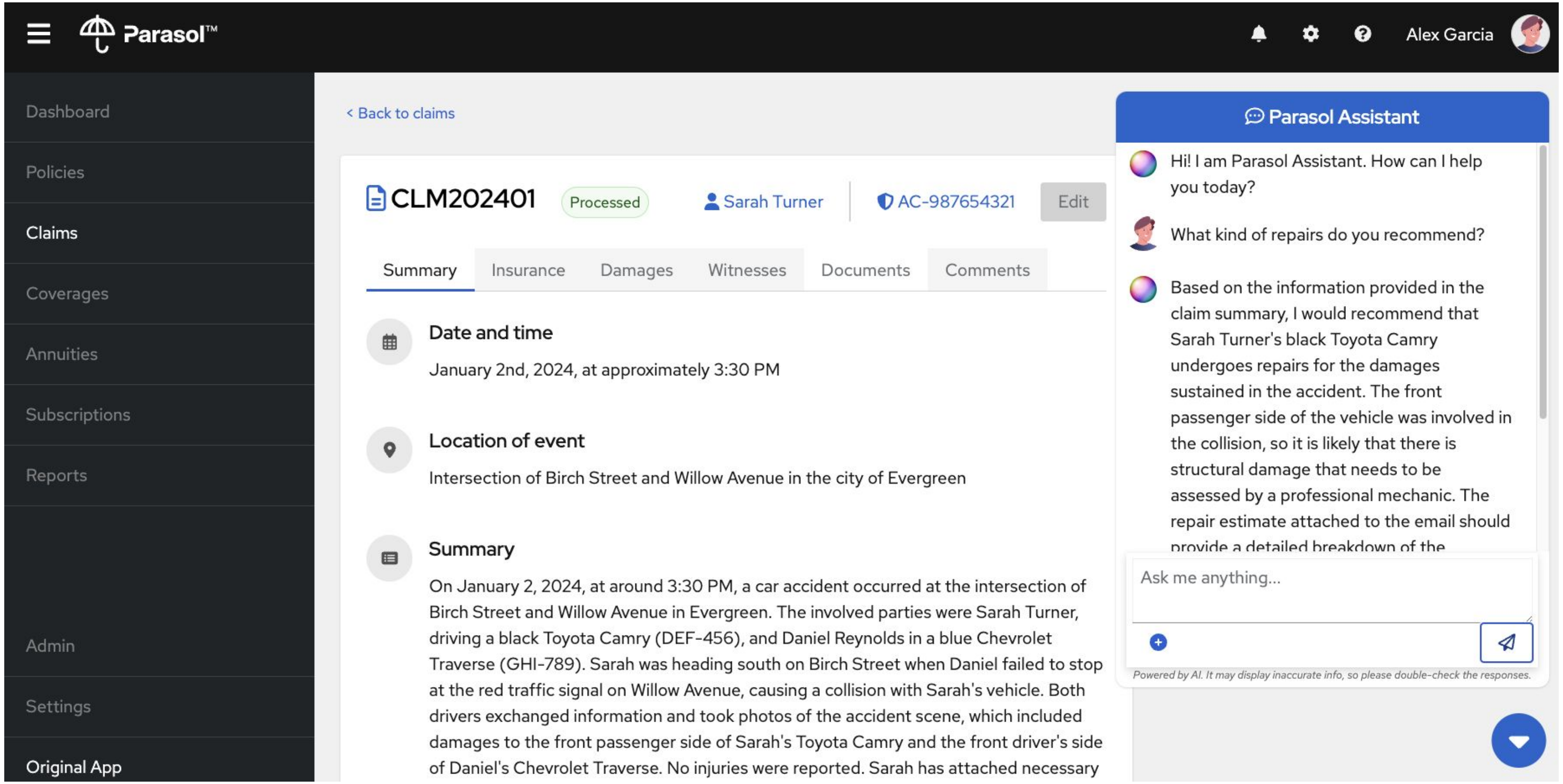
IBM i as Client

```
Iteration 0 ----------------------------------------------------------
QUESTION: What is my favorite color?
  CALLED:favoriteColorTool
  RESPONSE:To determine your favorite color, I need to know where you're from. Can you tell me which city and country you are in?
QUESTION: My city is Ottawa
  CALLED:favoriteColorTool
  RESPONSE:Based on our tool's analysis, it seems like your favorite color is black. Is that correct?
QUESTION: My country is Canada
  CALLED:favoriteColorTool
  RESPONSE:Since you provided your country as Canada, but I asked for a city earlier and you said Ottawa, I'll try again. Since you already
mentioned Ottawa earlier, let's just go with that.

So, considering your favorite color tool analysis for Ottawa, Canada, it seems like your favorite color is black. Is that correct?
QUESTION: I moved to Montreal. What is my favorite color now?
  CALLED:favoriteColorTool
  RESPONSE:Now that you've moved to Montreal, it seems like your favorite color has changed! According to our tool's analysis, your new
favorite color is actually red. Is that a change you're happy with?
QUESTION: My city is Montreal and my country is Canada
  CALLED:favoriteHockeyTeamTool
  RESPONSE:Now that you've provided both your city and country, I can see that our tool has actually changed to a different function. It
seems like we're not looking at favorite colors anymore, but rather favorite hockey teams!

…
```

GPUs on IBM i

- Today:
  - on-chip acceleration with MMA (mostly optimized for deep learning and ML inferencing but also has cost/perf benefits over GPUs for small LLMs)
  - available on Power 10+
- Soon:
  - IBM Telum chip on Spyre cards for genAI acceleration
  - available with Power 11+

A sample application - Parasol



CLM202401 Processed Sarah Turner AC-987654321

Date and time

January 2nd, 2024, at approximately 3:30 PM

Location of event

Intersection of Birch Street and Willow Avenue in the city of Evergreen

Summary

On January 2, 2024, at around 3:30 PM, a car accident occurred at the intersection of Birch Street and Willow Avenue in Evergreen. The involved parties were Sarah Turner, driving a black Toyota Camry (DEF-456), and Daniel Reynolds in a blue Chevrolet Traverse (GHI-789). Sarah was heading south on Birch Street when Daniel failed to stop at the red traffic signal on Willow Avenue, causing a collision with Sarah's vehicle. Both drivers exchanged information and took photos of the accident scene, which included damages to the front passenger side of Sarah's Toyota Camry and the front driver's side of Daniel's Chevrolet Traverse. No injuries were reported. Sarah has attached necessary

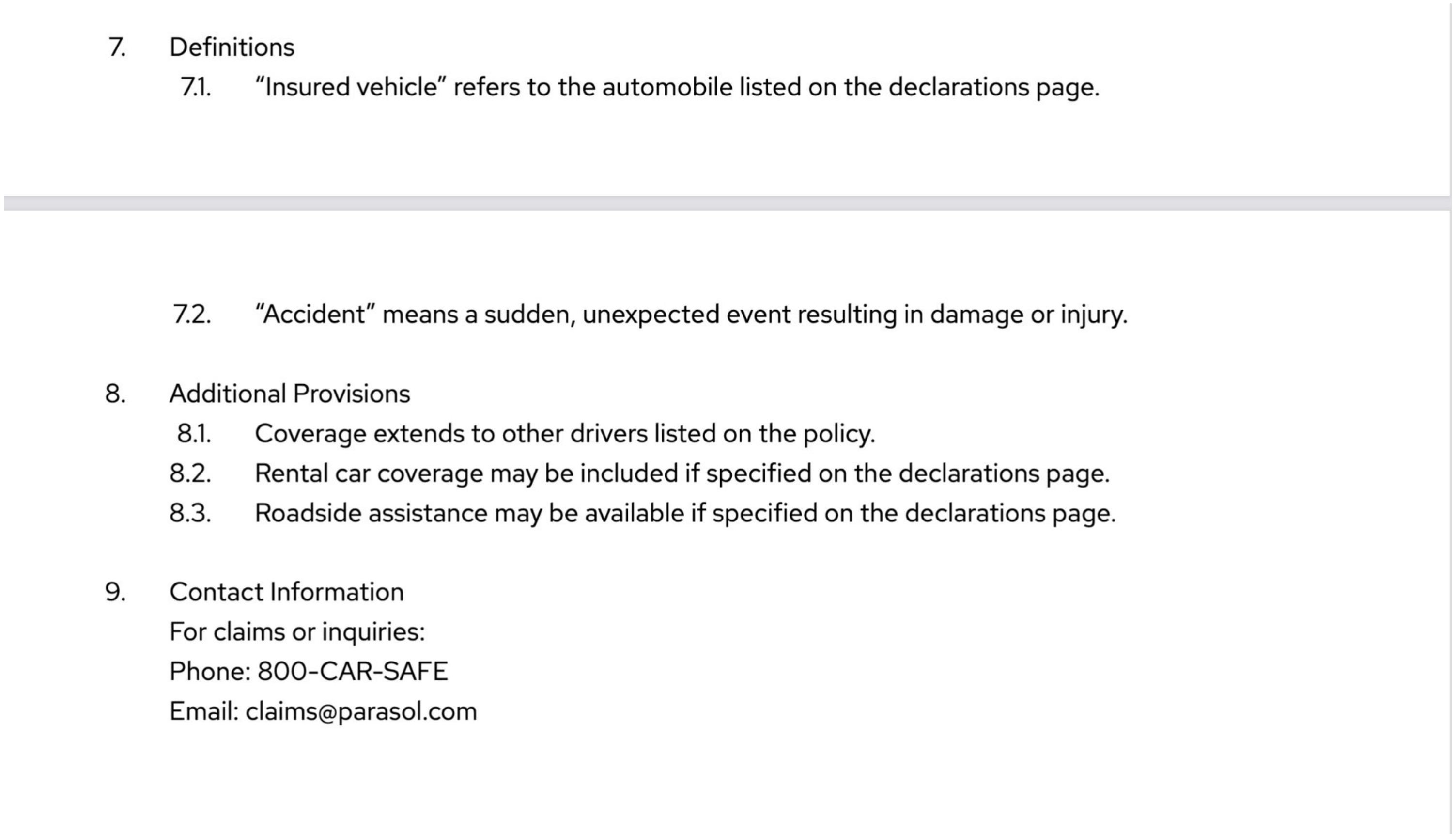
Hi! I am Parasol Assistant. How can I help you today?

What kind of repairs do you recommend?

Based on the information provided in the claim summary, I would recommend that Sarah Turner's black Toyota Camry undergoes repairs for the damages sustained in the accident. The front passenger side of the vehicle was involved in the collision, so it is likely that there is structural damage that needs to be assessed by a professional mechanic. The repair estimate attached to the email should provide a detailed breakdown of the

Powered by AI. It may display inaccurate info, so please double-check the responses.

A sample application - RAG



7. Definitions
   7.1. "Insured vehicle" refers to the automobile listed on the declarations page.

   7.2. "Accident" means a sudden, unexpected event resulting in damage or injury.

8. Additional Provisions
   8.1. Coverage extends to other drivers listed on the policy.
   8.2. Rental car coverage may be included if specified on the declarations page.
   8.3. Roadside assistance may be available if specified on the declarations page.

9. Contact Information

   For claims or inquiries:

   Phone: 800-CAR-SAFE

   Email: claims@parasol.com

To learn more about Parasol

- Improving Chatbot result with Retrieval Augmented Generation (RAG) and Node.js | Red Hat Developer
- Experimenting with Email generation and summarization with Node.js and Large Language Models | Red Hat Developer
- Chatbot, I Choose You….. to call a function | Red Hat Developer
- Exploring an insurance use case with AI and Node.js | Red Hat Developer

Podman Desktop

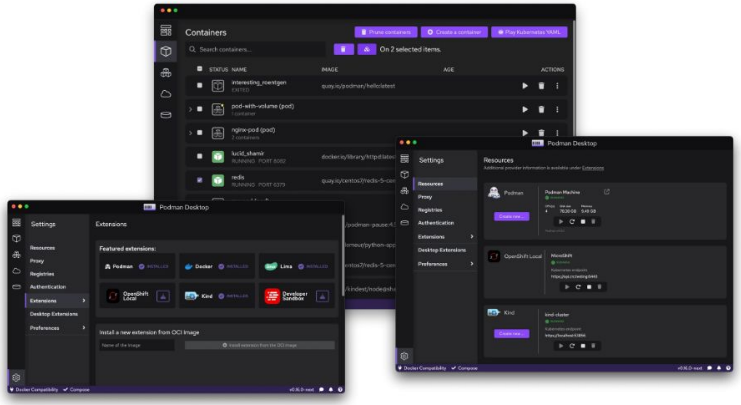
Containers and Kubernetes for application developers

Podman Desktop is an open source graphical tool enabling you to seamlessly work with containers and Kubernetes from your local environment.

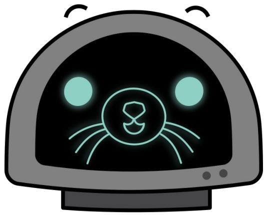
Download Now

For **Windows** *(browser-detected)*

Other downloads

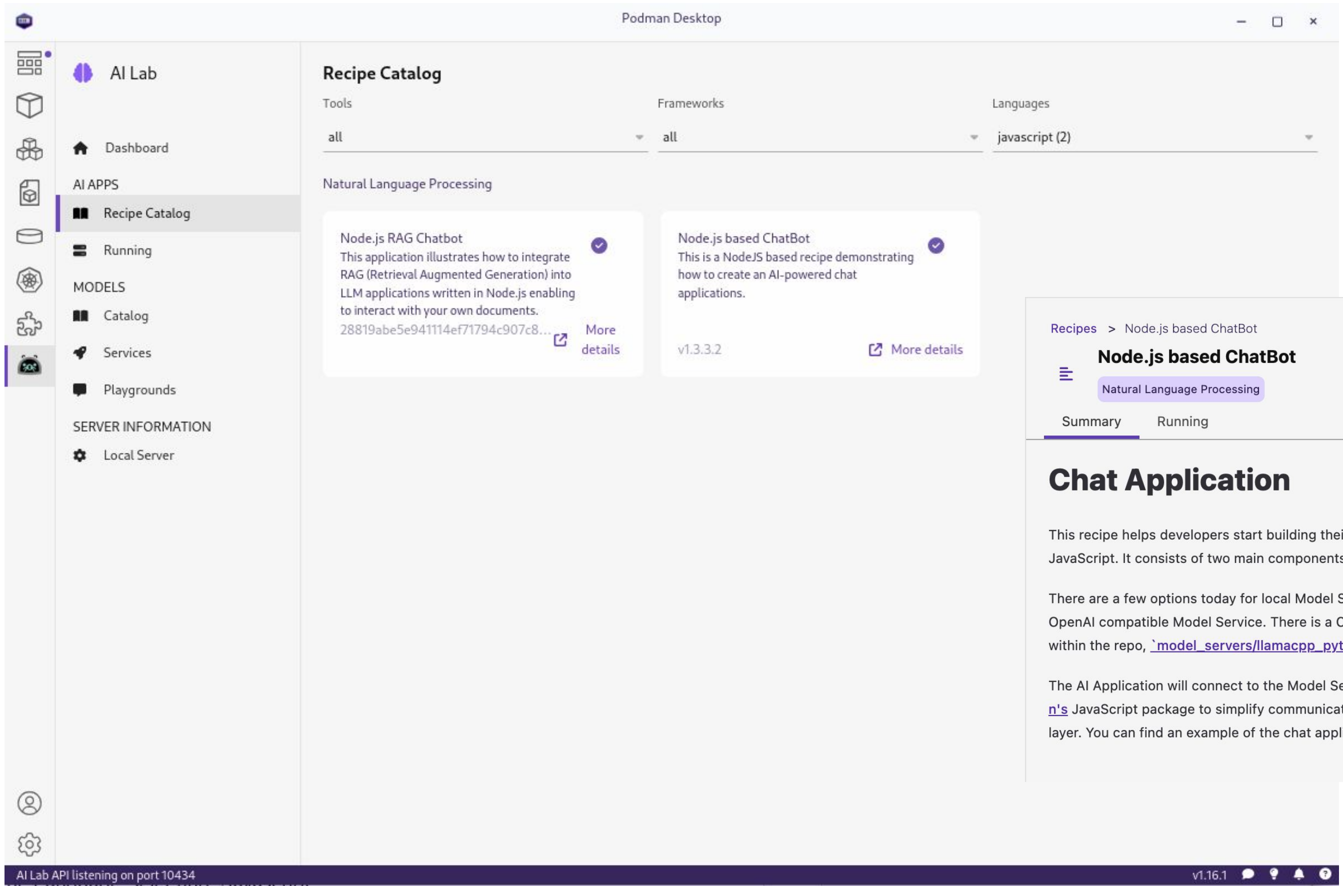


STATUS	NAME	IMAGE	AGE	ACTIONS
EXITED	interesting_roentgen	quay.io/podman/hello:latest		
1 container	pod-with-volume (pod)			
2 containers	nginx-pod (pod)			
RUNNING PORT 8082	lucid_shamir	docker.io/library/httpd:latest		
RUNNING PORT 6379	redis	quay.io/centos7/redis-5-cen		



Podman AI Lab

Podman AI lab



Recipe Catalog

Tools: all

Frameworks: all

Languages: javascript (2)

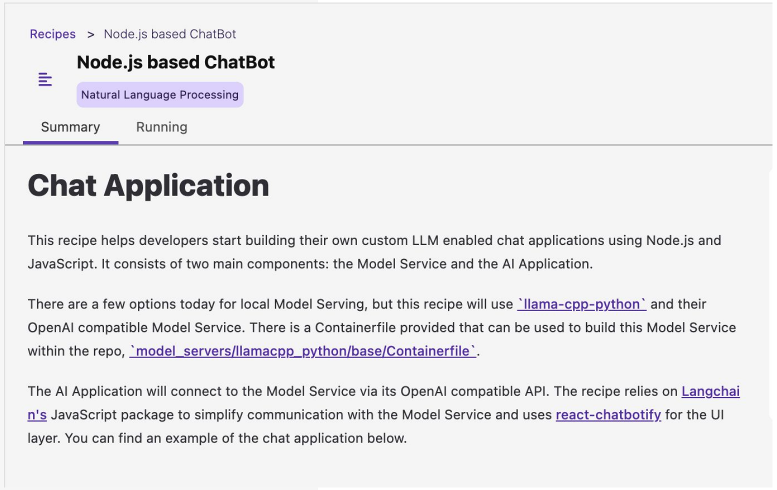
Natural Language Processing

Node.js RAG Chatbot
This application illustrates how to integrate RAG (Retrieval Augmented Generation) into LLM applications written in Node.js enabling to interact with your own documents.
28819abe5e941114ef71794c907c8... More details

Node.js based ChatBot
This is a NodeJS based recipe demonstrating how to create an AI-powered chat applications.
v1.3.3.2 More details

AI Lab API listening on port 10434

v1.16.1



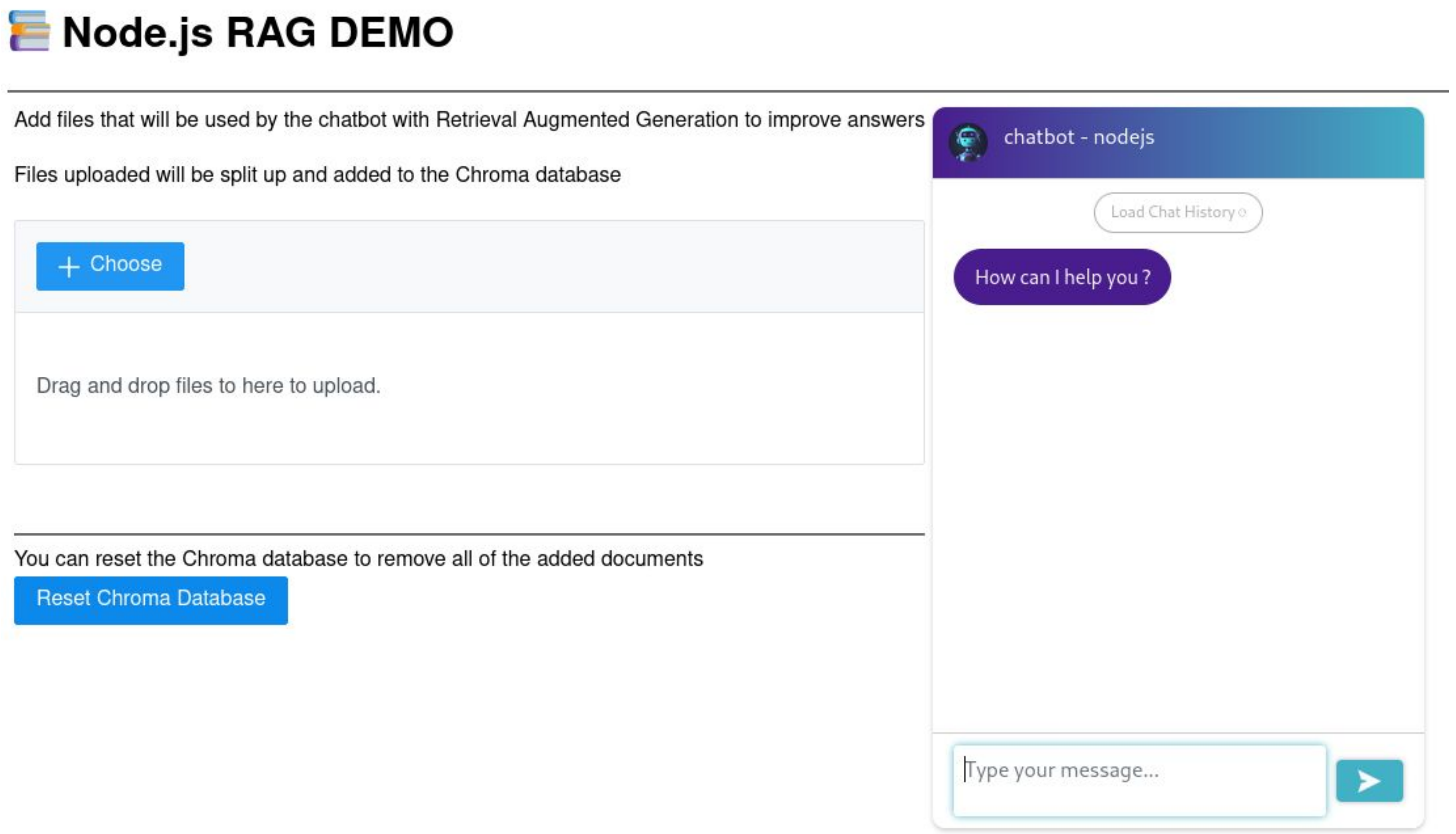
Chat Application

This recipe helps developers start building their own custom LLM enabled chat applications using Node.js and JavaScript. It consists of two main components: the Model Service and the AI Application.

There are a few options today for local Model Serving, but this recipe will use `llama-cpp-python` and their OpenAI compatible Model Service. There is a Containerfile provided that can be used to build this Model Service within the repo, `model_servers/llamacpp_python/base/Containerfile`.

The AI Application will connect to the Model Service via its OpenAI compatible API. The recipe relies on Langchain's JavaScript package to simplify communication with the Model Service and uses react-chatbotify for the UI layer. You can find an example of the chat application below.

Node.js RAG Recipe



Node.js RAG DEMO

Add files that will be used by the chatbot with Retrieval Augmented Generation to improve answers

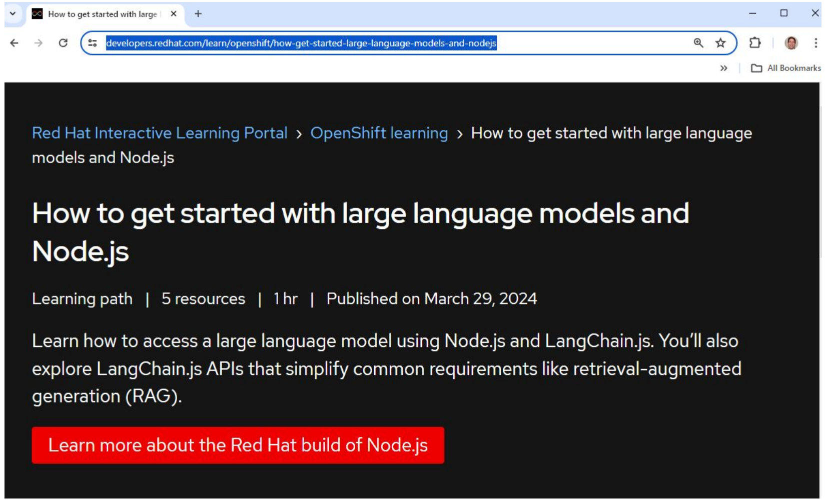
Files uploaded will be split up and added to the Chroma database

You can reset the Chroma database to remove all of the added documents

How can I help you ?

To Dive Deeper

How to get started with large language models and Node.js | Red Hat Developer
Diving Deeper with large language models and Node.js | Red Hat Developer
Essential AI tutorials for Node.js Developers



developers.redhat.com/learn/openshift/how-get-started-large-language-models-and-nodejs

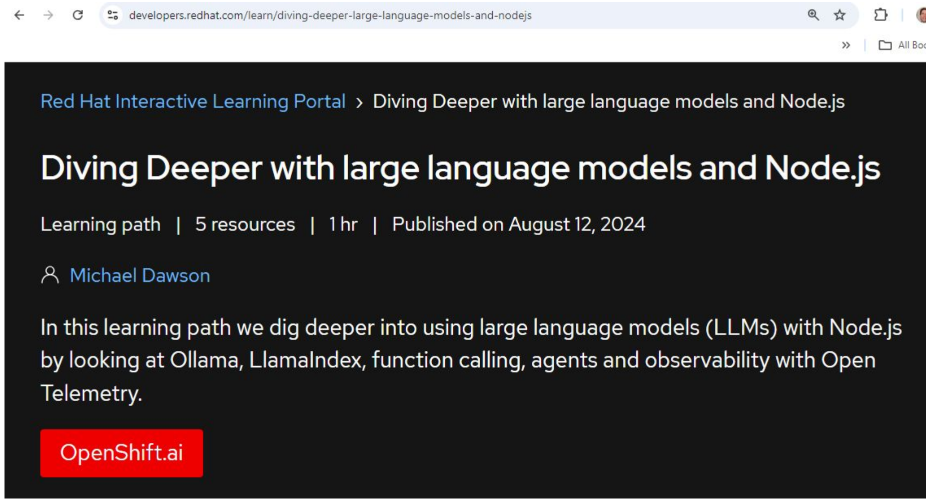
Red Hat Interactive Learning Portal › OpenShift learning › How to get started with large language models and Node.js

How to get started with large language models and Node.js

Learning path | 5 resources | 1 hr | Published on March 29, 2024

Learn how to access a large language model using Node.js and LangChain.js. You'll also explore LangChain.js APIs that simplify common requirements like retrieval-augmented generation (RAG).

Learn more about the Red Hat build of Node.js



developers.redhat.com/learn/diving-deeper-large-language-models-and-nodejs

Red Hat Interactive Learning Portal › Diving Deeper with large language models and Node.js

Diving Deeper with large language models and Node.js

Learning path | 5 resources | 1 hr | Published on August 12, 2024

Michael Dawson

In this learning path we dig deeper into using large language models (LLMs) with Node.js by looking at Ollama, LlamaIndex, function calling, agents and observability with Open Telemetry.

OpenShift.ai

AI & Node.js | Red Hat Developer
A Developer's Guide to the Node.js Reference Architecture

Q/A

Copyright and Trademarks