ChatGPT

# Langfuse-Ollama Custom Node for n8n

## Overview

This document outlines the development and configuration of a **custom Langfuse-Ollama node** for n8n, enabling secure, traceable interactions with Ollama LLMs while capturing observability data using Langfuse.

### Features

- Connects to **Ollama** securely using credentials
- Connects to **Langfuse** with credentials (public key, secret key, host)
- Dynamically fetches available **Ollama models**
- Accepts **prompt input** via Chat Trigger
- Creates and manages **Langfuse traces and spans**

---

## Node Structure

### File Path

```
~/.n8n/custom/nodes/LangfuseOllama.node.js
```

### Metadata

- `displayName`: Langfuse Ollama
- `name`: langfuseOllama
- `group`: transform
- `inputs`: main
- `outputs`: main

### Credentials Required

```
credentials: [
  { name: 'ollamaApi', required: true },
  { name: 'langfuseApi', required: true },
],
```

### Node Properties

```
properties: [
  {
    displayName: 'Prompt',
    name: 'prompt',
    type: 'string',
    default: 'What is the capital of Canada?',
  },
  {
    displayName: 'Model',
    name: 'model',
    type: 'options',
    typeOptions: {
      loadOptionsMethod: 'getOllamaModels',
    },
    default: '',
  },
  {
    displayName: 'Langfuse API',
    name: 'langfuseApi',
    type: 'credentials',
    credentials: ['langfuseApi'],
  },
  {
    displayName: 'Ollama API',
    name: 'ollamaApi',
    type: 'credentials',
    credentials: ['ollamaApi'],
  }
]
```

---

## Dynamic Model Loading

Implemented via:

```
async getOllamaModels() {
  const creds = await this.getCredentials('ollamaApi');
  const response = await this.helpers.request({
    method: 'GET',
    uri: `${creds.baseUrl}/api/tags`,
    json: true,
  });
  return (response.models || response.tags || []).map((model) => ({
```

```
    name: model.name || model,
    value: model.name || model,
  }));
}
```

---

## Credential Definitions

**1.** `OllamaApi.credentials.js`

```
class OllamaApi {
  constructor() {
    this.name = 'ollamaApi';
    this.displayName = 'Ollama API';
    this.credentials = [
      { name: 'baseUrl', type: 'string', default: 'http://localhost:11434' },
    ];
  }
}
module.exports = { OllamaApi };
```

**2.** `LangfuseApi.credentials.js`

```
class LangfuseApi {
  constructor() {
    this.name = 'langfuseApi';
    this.displayName = 'Langfuse API';
    this.credentials = [
      { name: 'publicKey', type: 'string', default: '' },
      { name: 'secretKey', type: 'string', default: '' },
      { name: 'baseUrl', type: 'string', default: 'http://localhost:3001' },
    ];
  }
}
module.exports = { LangfuseApi };
```

---

## Execution Logic

### LLM + Langfuse Tracing Flow

```
const langfuse = new Langfuse({ publicKey, secretKey, baseUrl, debug: true });
const trace = langfuse.trace({ name: 'ollama-node-trace', userId: 'n8n-langfuse-node' });
const span = trace.span({ name: 'ollama-inference' });

const llm = new Ollama({ model, baseUrl: ollamaCreds.baseUrl });
const response = await llm.invoke(prompt);

await span.update({ input: prompt, output: response });
await span.end();
await trace.update({ input: prompt, output: response });
await langfuse.flush();
```

---

## Docker Dev Environment

Use the following containers: - **n8n** (with mounted custom node/creds) - **Ollama** (`mistral` model pulled) - **Langfuse** (`langfuse/langfuse:2`) + PostgreSQL

Ensure networking and `baseUrl` match internal host (e.g., `host.docker.internal`).

---

## Next Steps

- Integrate with `LLMChain` or custom AI agent workflow
- Add retry/backoff handling
- Optional: UI improvements for credential selection labels

---

## Summary

This Langfuse-Ollama node brings production-ready observability to local LLMs via secure, dynamic, and traceable interactions. Ideal for building intelligent, debuggable agents within n8n.

For questions, integration help, or improvements, contact the developer or project maintainer.