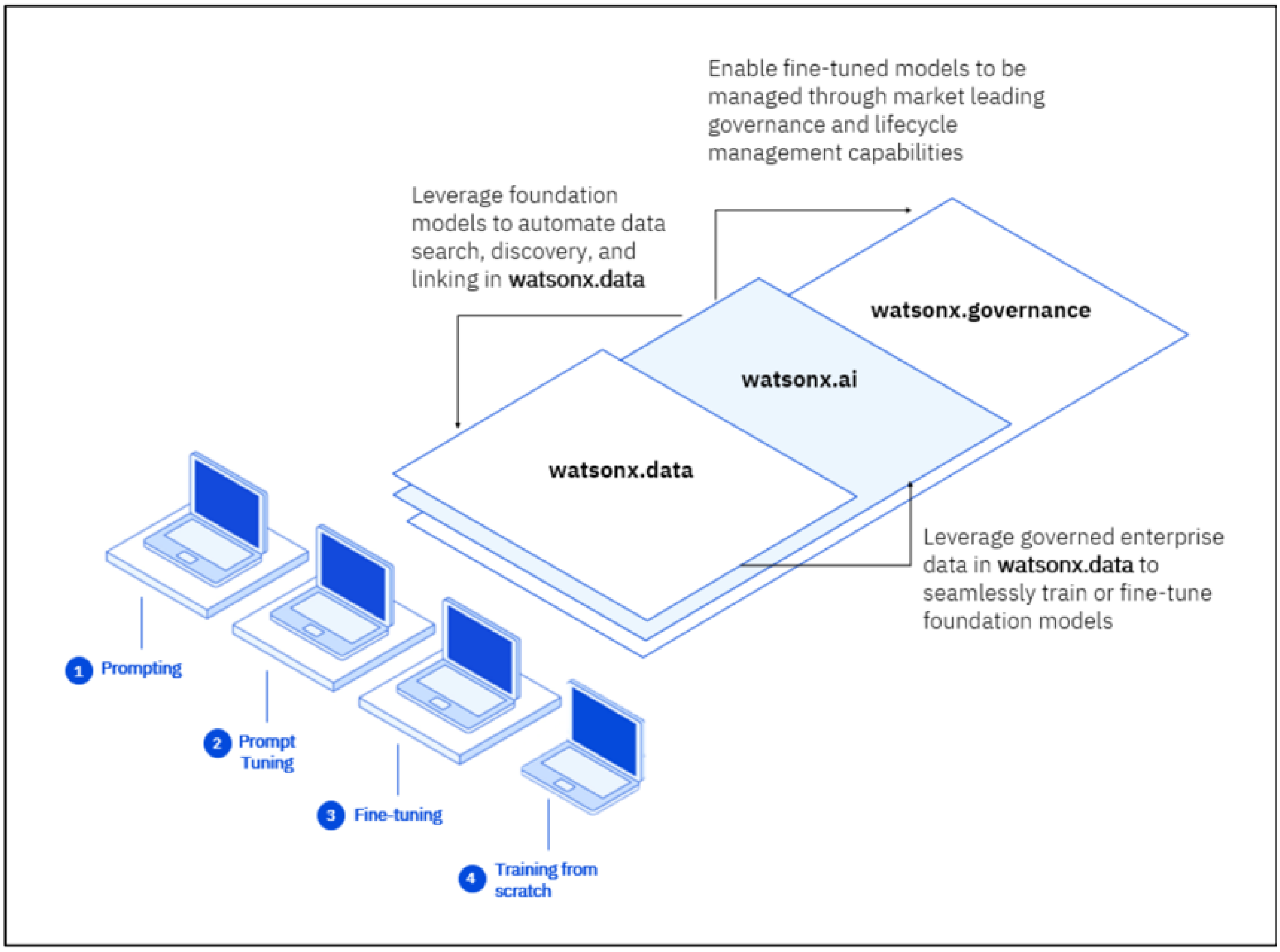
Creating Agents with IBM watsonx.ai and BeeAI

Contents

1 Introduction

IBM watsonx.ai is a core component of watsonx, IBM's enterprise-ready AI and data platform that's designed to multiply the impact of AI across an enterprise.

The watsonx platform has three powerful components: the watsonx.ai studio for new foundation models, generative AI, and Machine Learning (traditional AI); the watsonx.data fit-for-purpose data lakehouse that provides the flexibility of a data lake with the performance of a data warehouse; and the watsonx.governance toolkit, which enables AI workflows that are built with responsibility, transparency, and explainability.


Enable fine-tuned models to be managed through market leading governance and lifecycle management capabilities
Leverage foundation models to automate data search, discovery, and linking in watsonx.data
watsonx.governance
watsonx.ai
watsonx.data
Leverage governed enterprise data in watsonx.data to seamlessly train or fine-tune foundation models
1 Prompting
2 Prompt Tuning
3 Fine-tuning
4 Training from scratch

The watsonx.ai component (the focus of this lab) makes it possible for enterprises to train, validate, tune, and deploy AI models - both traditional AI and generative AI. With watsonx.ai, enterprises can leverage their existing traditional AI investment as well as exploit the innovations and potential of generative AI builds on foundation models to bring advanced automation and AI-infused applications to reduce cost, improve efficiency, scale, and accelerate the impact of AI across their organizations.

1.1 Introduction to BeeAI

BeeAI is a framework for building production-ready multi-agent systems in **Python** or **TypeScript**.

Why BeeAI?

- 🏆 **Build for your use case**. Implement simple to complex multi-agent patterns using Workflows, start with a ReActAgent, or easily build your own agent architecture. There is no one-size-fits-all agent architecture, you need full flexibility in orchestrating agents and defining their roles and behaviors.

- 🔌 **Seamlessly integrate with your models and tools**. GGet started with any model from Ollama, Groq, OpenAI, watsonx.ai, and more. Leverage tools from LangChain, connect to any server using the Model Context Protocol, or build your own custom tools. BeeAI is designed to integrate with the systems and capabilities you need.

- 🚀 **Scale with production-grade controls**. Optimize token usage through memory strategies, persist and restore agent state via (de)serialization, generate structured outputs, and execute generated code in a sandboxed environment. When things go wrong, BeeAI tracks the full agent workflow through events, collects telemetry, logs diagnostic data, and handles errors with clear, well-defined exceptions. Deploying multi-agent systems requires resource management and reliability.

1.2 About this Lab

This lab provides hands-on experience developing AI agents using watsonx.ai and the BeeAI framework.

2 Prerequisites

Technical Requirements

- JavaScript runtime NodeJS > 18 (ideally installed via nvm).
- Container system like Rancher Desktop or Podman Desktop (VM must be rootfull machine).
    - **Note**: this is only required if you want to leverage the code interpreter and/or the observability features.
- LLM Provider either external WatsonX (OpenAI, Groq, ...) or local ollama.

watsonx.ai Environment

You must have access to a watsonx.ai SaaS environment and an initialized project within that environment. If you do not have one already, it can be provisioned on TechZone by selecting the **watsonx.ai/.governance SaaS** environment and selecting **Education** as **Purpose**.

To run this lab, you will need:

- An IBM Cloud API Key.
- Your watsonx.ai SaaS URL and project ID.

3 Lab Preparation

3.1 Create the Development Environment

All lab development will be performed in a single Python project tree. The steps below should work on all platforms.

3.1.1 Project Folder

To contain all the lab exercises, create a parent working folder:

```
mkdir beeai-lab
```

4 Get started with BeeAI

Step 1: Chat with your first BeeAI agents

Navigate to your lab folder:

```
cd beeai-lab
```

1. Clone the BeeAI starter repository:

```
git clone https://github.com/i-am-bee/beeai-framework-starter
cd beeai-framework-starter
```

2. Install the dependencies using `npm ci`.

```
npm ci
```

3. Configure your project by filling in missing values in the `.env` file, like the following:

```
LLM_CHAT_MODEL_NAME="watsonx"

...OMITTED...

# For Watsonx LLM Adapter
WATSONX_CHAT_MODEL="ibm/granite-3-8b-instruct"
WATSONX_EMBEDDING_MODEL="ibm/slate-125m-english-rtrvr"
WATSONX_API_KEY="<CHANGEME>"
WATSONX_PROJECT_ID="<CHANGEME>"
WATSONX_VERSION="2023-05-29"
WATSONX_REGION="eu-de"
```

4. Run the agent using `npm run start src/agent.ts`.

```
npm run start src/agent.ts
```

5. To run an agent with custom prompt, simply do this:

```
npm run start src/agent.ts <<< 'Hello Bee!'
```

Step 2: Explore the Code Interpreter

The Bee Code Interpreter is a gRPC service an agent uses to execute an arbitrary Python code safely.

Instructions

1. Start all services related to the `Code Interpreter`:

```
npm run infra:start --profile=code_interpreter
```

2. Run the agent:

```
npm run start src/agent_code_interpreter.ts
```

3. You can now use prebuilt tools in the `src/agent_code_interpreter.ts` that use the code interpreter, by asking something like:

```
Give me a riddle
```

   - Sample output:

```
Agent 🤖 (thought) :  To provide a riddle, I need to fetch one from a
source, in this case, I can use the get_riddle function to retrieve a
random riddle and its answer.
Agent 🤖 (tool_name) :  get_riddle
Agent 🤖 (tool_input) :  {}
Agent 🤖 (tool_output) :  {"riddle": "What is round as a dishpan,
deep as a tub, and still the oceans couldn't fill it up?", "answer":
"A sieve"}
Agent 🤖 (thought) :  The function output contains the riddle and its
answer, now I can provide the riddle to the user.
Agent 🤖 (final_answer) :  Here's a riddle for you: What is round as
a dishpan, deep as a tub, and still the oceans couldn't fill it up?
Think you can solve it?
Agent 🤖 :  Here's a riddle for you: What is round as a dishpan, deep
as a tub, and still the oceans couldn't fill it up? Think you can
solve it?
```

4. You can also ask complexe queries that require calculations/code to be defined and ran by your agent, like:

```
What are the monthly repayments on a $948,000 loan at an interest rate
of 5.85% over 30 years
```

   - Sample output:

```
Agent 🤖 (thought) :  To calculate the monthly repayments on a loan,
we can use a financial formula, which can be implemented in Python.
...OMITTED...
```

```
Agent 🤖 :  The monthly repayment on a $948,000 loan at an interest
rate of 5.85% over 30 years is approximately $5,592.64.
```

Step 3: Explore Observability

Get complete visibility of the agent's inner workings via our observability stack.

- MLFlow is used as UI for observability.
- The Bee Observe is the observability service (API) for gathering traces from Bee Agent Framework.

Instructions

1. Start all services related to Bee Observe:

```
npm run infra:start --profile=observe
```

2. Run the agent:

```
npm run start src/agent_observe.ts
```

3. Ask something like:

```
What is the current weather in Las Vegas?
```

   - Sample output:

```
User 👤 : What is the current weather in Las Vegas?
Agent 🤖 :  The current weather in Las Vegas is 11.5°C with a
relative humidity of 72% and a wind speed of 15.3 km/h.
```

4. Visualize the trace in the MLFlow web application: `http://127.0.0.1:8080/#/experiments/0`

Step 4: Run a multi-agent-workflow

Now let's run a multi-agent workflow to help us generate curated content on a given topic.

Instructions

1. Run the multi-agent workflow:

```
npm run src/agent_workflow.ts
```

2. Ask something like:

```
Help me create a blog post around AI agents
```

   - Sample output:

```
User 👤 : Help me create a blog post around AI agents
-> ▶ preprocess {"input":"Help me create a blog post around AI
agents","notes":[]}
-> ▶ planner {"input":"Help me create a blog post around AI
agents","topic":"AI agents","notes":[]}
-> ▶ writer {"input":"Help me create a blog post around AI
agents","topic":"AI agents",...
-> ▶ editor {"input":"Help me create a blog post around AI
agents","topic":"AI agents","notes":[],"plan":"\n# Content ...
🤖 Answer Introduction to AI Agents: Revolutionizing Business and
Automation...
```

3. **Note** how `Content Planner`, `Content Writer` and `Editor` agents are working together to provide you with the response. Navigate to the `src/agent_workflow.ts` to see how the workflow has been implemented.

Conclusion

In this hands-on lab, we explored the BeeAI Framework and its key features, including the Code Interpreter and Observability. We also covered the step-by-step instructions for getting started with the framework. With this knowledge, you can now start building and deploying your own AI-powered agents the BeeAI Framework.