# pydantic_ai.models.ollama

## Setup

For details on how to set up authentication with this model, see model configuration for Ollama.

## Example local usage

With `ollama` installed, you can run the server with the model you want to use:

**terminal-run-ollama**

```
ollama run llama3.2
```

(this will pull the `llama3.2` model if you don't already have it downloaded)

Then run your code, here's a minimal example:

**ollama_example.py**

```
from pydantic import BaseModel

from pydantic_ai import Agent


class CityLocation(BaseModel):
    city: str
    country: str


agent = Agent('ollama:llama3.2', result_type=CityLocation)

result = agent.run_sync('Where the olympics held in 2012?')
print(result.data)
#> city='London' country='United Kingdom'
print(result.cost())
#> Cost(request_tokens=56, response_tokens=8, total_tokens=64, details=None)
```

## Example using a remove server

**ollama_example_with_remote_server.py**

```
from pydantic import BaseModel

from pydantic_ai import Agent
from pydantic_ai.models.ollama import OllamaModel

ollama_model = OllamaModel(
    model_name='qwen2.5-coder:7b',  # (1)
    base_url='http://192.168.1.74:11434/v1',  # (2)
)


class CityLocation(BaseModel):
    city: str
    country: str


agent = Agent(model=ollama_model, result_type=CityLocation)

result = agent.run_sync('Where the olympics held in 2012?')
print(result.data)
#> city='London' country='United Kingdom'
print(result.cost())
#> Cost(request_tokens=56, response_tokens=8, total_tokens=64, details=None)
```

1. The name of the model running on the remote server
2. The url of the remote server

See `OllamaModel` for more information

### CommonOllamaModelNames `module-attribute`

```
CommonOllamaModelNames = Literal[
    "codellama",
    "gemma",
    "gemma2",
    "llama3",
    "llama3.1",
    "llama3.2",
    "llama3.2-vision",
    "llama3.3",
    "mistral",
    "mistral-nemo",
    "mixtral",
    "phi3",
    "qwq",
    "qwen",
    "qwen2",
    "qwen2.5",
    "starcoder2",
]
```

This contains just the most common ollama models.

For a full list see ollama.com/library.

### OllamaModelName `module-attribute`

```
OllamaModelName = Union[CommonOllamaModelNames, str]
```

Possible ollama models.

Since Ollama supports hundreds of models, we explicitly list the most models but allow any name in the type hints.

## OllamaModel `dataclass`

Bases: `Model`

A model that implements Ollama using the OpenAI API.

Internally, this uses the OpenAI Python client to interact with the Ollama server.

Apart from `__init__`, all methods are private or match those of the base class.

**Source code in `pydantic_ai_slim/pydantic_ai/models/ollama.py`**

```python
@dataclass(init=False)
class OllamaModel(Model):
    """A model that implements Ollama using the OpenAI API.

    Internally, this uses the [OpenAI Python client](https://github.com/openai/openai-python) to interact with the Ollama server.

    Apart from `__init__`, all methods are private or match those of the base class.
    """

    model_name: OllamaModelName
    openai_model: OpenAIModel

    def __init__(
        self,
        model_name: OllamaModelName,
        *,
        base_url: str | None = 'http://localhost:11434/v1/',
        openai_client: AsyncOpenAI | None = None,
        http_client: AsyncHTTPClient | None = None,
    ):
        """Initialize an Ollama model.

        Ollama has built-in compatability for the OpenAI chat completions API ([source](https://ollama.com/blog/openai-compatibility)), so we reuse the
        [`OpenAIModel`][pydantic_ai.models.openai.OpenAIModel] here.

        Args:
            model_name: The name of the Ollama model to use. List of models available [here](https://ollama.com/library)
                You must first download the model (`ollama pull <MODEL-NAME>`) in order to use the model
            base_url: The base url for the ollama requests. The default value is the ollama default
            openai_client: An existing
                [`AsyncOpenAI`](https://github.com/openai/openai-python?tab=readme-ov-file#async-usage)
                client to use, if provided, `base_url` and `http_client` must be `None`.
            http_client: An existing `httpx.AsyncClient` to use for making HTTP requests.
        """
        self.model_name = model_name
        if openai_client is not None:
            assert base_url is None, 'Cannot provide both `openai_client` and `base_url`'
            assert http_client is None, 'Cannot provide both `openai_client` and `http_client`'
            self.openai_model = OpenAIModel(model_name=model_name, openai_client=openai_client)
        else:
            # API key is not required for ollama but a value is required to create the client
            http_client_ = http_client or cached_async_http_client()
            oai_client = AsyncOpenAI(base_url=base_url, api_key='ollama', http_client=http_client_)
            self.openai_model = OpenAIModel(model_name=model_name, openai_client=oai_client)

    async def agent_model(
        self,
        *,
        function_tools: list[ToolDefinition],
        allow_text_result: bool,
        result_tools: list[ToolDefinition],
    ) -> AgentModel:
        return await self.openai_model.agent_model(
            function_tools=function_tools,
            allow_text_result=allow_text_result,
            result_tools=result_tools,
        )

    def name(self) -> str:
        return f'ollama:{self.model_name}'
```

**__init__**

```python
__init__(
    model_name: OllamaModelName,
    *,
    base_url: str | None = "http://localhost:11434/v1/",
    openai_client: AsyncOpenAI | None = None,
    http_client: AsyncClient | None = None
)
```

Initialize an Ollama model.

Ollama has built-in compatability for the OpenAI chat completions API (source), so we reuse the `OpenAIModel` here.

**Parameters:**

| Name | Type | Description | Default |
|---|---|---|---|
| `model_name` | `OllamaModelName` | The name of the Ollama model to use. List of models available here You must first download the model (`ollama pull <MODEL-NAME>`) in order to use the model | *required* |
| `base_url` | `str \| None` | The base url for the ollama requests. The default value is the ollama default | `'http://localhost:11434/v1/'` |
| `openai_client` | `AsyncOpenAI \| None` | An existing `AsyncOpenAI` client to use, if provided, `base_url` and `http_client` must be `None`. | `None` |
| `http_client` | `AsyncClient \| None` | An existing `httpx.AsyncClient` to use for making HTTP requests. | `None` |

**Source code in** `pydantic_ai_slim/pydantic_ai/models/ollama.py`

```python
def __init__(
    self,
    model_name: OllamaModelName,
    *,
    base_url: str | None = 'http://localhost:11434/v1/',
    openai_client: AsyncOpenAI | None = None,
    http_client: AsyncHTTPClient | None = None,
):
    """Initialize an Ollama model.

    Ollama has built-in compatability for the OpenAI chat completions API ([source](https://ollama.com/blog/openai-compatibility)), so we reuse the
    [`OpenAIModel`][pydantic_ai.models.openai.OpenAIModel] here.

    Args:
        model_name: The name of the Ollama model to use. List of models available [here](https://ollama.com/library)
            You must first download the model (`ollama pull <MODEL-NAME>`) in order to use the model
        base_url: The base url for the ollama requests. The default value is the ollama default
        openai_client: An existing
            [`AsyncOpenAI`](https://github.com/openai/openai-python?tab=readme-ov-file#async-usage)
            client to use, if provided, `base_url` and `http_client` must be `None`.
        http_client: An existing `httpx.AsyncClient` to use for making HTTP requests.
    """
    self.model_name = model_name
    if openai_client is not None:
        assert base_url is None, 'Cannot provide both `openai_client` and `base_url`'
        assert http_client is None, 'Cannot provide both `openai_client` and `http_client`'
        self.openai_model = OpenAIModel(model_name=model_name, openai_client=openai_client)
    else:
        # API key is not required for ollama but a value is required to create the client
        http_client_ = http_client or cached_async_http_client()
        oai_client = AsyncOpenAI(base_url=base_url, api_key='ollama', http_client=http_client_)
        self.openai_model = OpenAIModel(model_name=model_name, openai_client=oai_client)
```