# Running Llama 3.2 Locally with Ollama and Docker: A Step-by-Step Guide

In this guide, I'll walk you through the process of deploying Meta's Llama 3.2 language model using

Ollama and Docker. This setup provides a straightforward way to run powerful AI models on your local

machine with minimal configuration.

**What is Ollama?**

Ollama is an open-source tool that simplifies running large language models locally. It handles the

complex process of downloading, setting up, and running these models with a user-friendly interface. By combining Ollama with Docker, we can create an isolated, portable environment for our AI applications.

## Prerequisites

Before we begin, you'll need:

* A machine with sufficient RAM (at least 8GB recommended)
* Docker installed
* Basic familiarity with command line interfaces

### Step 1: Install Docker

First, you need to have Docker installed on your system. Docker allows you to package and run

applications in isolated containers.

If you haven't installed Docker yet, you can download it from the official Docker website and follow the

installation instructions for your operating system.

After installation, verify Docker is running correctly with:

$ docker --version

This image includes the vLLM inference engine with an API compatible with OpenAI's endpoints.

### Step 2: Pull the Ollama Docker Image

Ollama provides an official Docker image that we can use. Pull the latest version with:

$ docker pull ollama/ollama

This command downloads the Ollama image from Docker Hub to your local machine.

### Step 3: Deploy Ollama in a Docker Container

$ docker run -d -v ollama:/root/.ollama -p 11434:11434 --name ollama ollama/ollama

Let's break down this command:

* -**d** : Runs the container in detached mode (in the background)
* **-v ollama:/root/.ollama** : Creates a Docker volume named "ollama" and mounts it to the

container's /root/.ollama directory, which persists model data between container restarts

* **-p 11434:11434** : Maps port 11434 from the container to your host, enabling API access
* **--name ollama** : Assigns the name "ollama" to the container for easy reference
* **ollama/ollama** : Specifies the Docker image to use

After running this command, Ollama will be running as a service on your machine, accessible via port

11434.

### Step 4: Access the Ollama Container

To interact directly with the Ollama container, execute:

$ docker exec -it ollama bash

This command gives you a bash shell inside the running Ollama container, where you can run Ollama

commands directly.

### Step 5: Download and Run the Llama 3.2 Model

Now that you're inside the container, you can download and run the Llama 3.2 model:

First, pull the model:

$ ollama pull llama3.2:1b

This downloads the 1 billion parameter version of Llama 3.2, which is smaller and requires less RAM than larger versions.

Then, run the model:

$ ollama run llama3.2:1b

This launches an interactive session where you can chat directly with the Llama 3.2 model. Type your

prompts and the model will respond in real-time.

To exit the interactive session, press **Ctrl+D** or type **/exit** .

### Step 6: Restart the Ollama Container (When needed)

If you need to restart the Ollama container for any reason, you can use:

$ docker restart ollama

This restarts the container without losing your downloaded models, as they're stored in the persistent

Docker volume we created earlier.

### Step 7: Test the API with cURL

You can also interact with your locally running model through Ollama's API. Here's how to test it with

cURL:

$ curl -X POST -d '{"prompt": "Tell me about AWS.", "model": "llama3.2:1b", "stream": false}' -H "Content-Type: application/json" http:<host-ip>:11434/api/generate

Replace **localhost** with your host's IP address if you're running this from another machine.

This command sends a request to the Ollama API asking for information about AWS. The **"stream":**

**false** parameter tells Ollama to return the complete response at once rather than streaming it token by token.

This command sends a request to the completion’s endpoint with:

* The model identifier
* A system prompt
* Temperature setting of 0.7 (controls randomness)
* No top-k filtering
* Maximum response length of 500 tokens

If everything is working correctly, you should receive a response with a joke from the model.

### Advanced Usage

#### Running Different Models

Ollama supports many different models. To see available models:

$ ollama list

To pull a different model:

$ ollama pull mistral

#### API Integration

You can integrate Ollama with your applications using its REST API. The API endpoint is available at **http://localhost:11434/api/generate** and accepts POST requests with JSON payloads.

### Conclusion

You've now successfully set up Llama 3.2 running locally using Ollama and Docker. This setup gives you:

* Local AI capabilities without dependency on cloud services
* Complete privacy for your data and prompts
* No API usage costs
* Flexibility to run different models as needed

This approach is ideal for development, testing, offline usage, or any situation where you want to keep your data local while still leveraging powerful AI capabilities.

Happy prompting with your local Llama 3.2 model!