



# Running Ollama with Open WebUI on Intel Hardware Platform

Installation Guide

---

*June 2024*



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visiting the [Intel Resource and Documentation Center](#).

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com](#).

No product or component can be absolutely secure.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

# Contents

---

<b>1.0</b>	<b>Introduction .....</b>	<b>5</b>
1.1	Verified Hardware.....	5
<b>2.0</b>	<b>Windows 11 Installation Guide.....</b>	<b>6</b>
2.1	Step 1: Installing Intel GPU Driver and oneAPI Base Toolkit (Windows) .....	6
2.1.1	Installing GPU Driver .....	6
2.1.2	Installing Intel oneAPI Base Toolkit.....	7
2.2	Step 2: Installing Python 3.11 (Windows).....	11
2.2.1	Successful Installation.....	13
2.3	Step 3: Installing IPEX-LLM for Ollama (Windows).....	14
2.3.1	Installing and Initializing Ollama with Intel GPU.....	14
2.3.2	Run Ollama Serve with Intel GPU .....	14
2.3.3	Successful Installation.....	15
2.4	Step 4: Installing Open WebUI (Windows).....	16
2.4.1	Install Open WebUI.....	16
2.4.2	Start Open WebUI Server.....	16
2.4.3	Successful Installation.....	16
<b>3.0</b>	<b>Ubuntu Installation Guide .....</b>	<b>17</b>
3.1	Step 1: Installing Intel GPU Driver and oneAPI Base Toolkit (Ubuntu).....	17
3.1.1	Installing GPU Driver .....	17
3.1.2	Installing Intel oneAPI Base Toolkit.....	17
3.2	Step 2: Installing Python 3.11 (Ubuntu) .....	20
3.2.1	Update System Packages.....	20
3.2.2	Adding PPA Repository.....	20
3.2.3	Python 3.11 Installation.....	20
3.2.4	Validate Python Version.....	21
3.3	Step 3: Installing IPEX-LLM for Ollama (Ubuntu) .....	21
3.3.1	Installing and initializing Ollama with Intel GPU.....	21
3.3.2	Run Ollama Serve with Intel GPU .....	21
3.3.3	Successful Installation.....	22
3.4	Step 4: Installing Open WebUI (Ubuntu).....	22
3.4.1	Install Open WebUI.....	22
3.4.2	Start Open WebUI Server.....	22
3.4.3	Successful Installation.....	22
<b>4.0</b>	<b>Configuring Open WebUI with Ollama.....</b>	<b>24</b>
4.1	Step 5: Configuring Open WebUI with Ollama.....	24
4.1.1	Setting Up Ollama with Open WebUI .....	24
4.2	Download Model and Inferencing.....	26

## Figures

Figure 1: Intel® Arc™ Graphics GPU Driver Official Download Page.....	6
Figure 2: Intel® Arc™ Graphics GPU Driver Installation.....	7
Figure 3: Intel® Arc™ Graphics Driver Dashboard.....	7
Figure 4: Intel oneAPI Base Toolkit Download Page.....	8
Figure 5: Installation of Intel oneAPI Base Toolkit.....	8
Figure 6: Installation of Intel oneAPI Base Toolkit.....	9
Figure 7: Installation of Intel oneAPI Base Toolkit.....	9
Figure 8: Successful Installation of Intel oneAPI Base Toolkit.....	10
Figure 9: Python 3.11 Official Download Page.....	11
Figure 10: Python 3.11 Installation Process.....	11
Figure 11: Python 3.11 Installation Process.....	12
Figure 12: Python 3.11 Installation Process.....	12
Figure 13: Steps on Opening Command Prompt as Administrator.....	13
Figure 14: Command Prompt Output to Ensure Successful Installation of Python 3.11.....	13
Figure 15: Successful Installation of Ollama Service.....	15
Figure 16: Successful Installation of Open WebUI.....	16
Figure 17: GPU Driver Installation Page.....	17
Figure 18: Intel oneAPI Base Toolkit Download Page.....	18
Figure 19: Intel oneAPI Base Toolkit Command Line Guide.....	18
Figure 20: Installation of Intel oneAPI Base Toolkit.....	19
Figure 21: Installation of Intel oneAPI Base Toolkit.....	19
Figure 22: Successful Installation of Intel oneAPI Base Toolkit.....	20
Figure 23: Successful Installation of Ollama Service.....	22
Figure 24: Successful Installation of Open WebUI.....	23
Figure 25: Step 1 of Setting Up Ollama with Open WebUI.....	24
Figure 26: Step 2 of Setting Up Ollama with Open WebUI.....	24
Figure 27: Step 3 of Setting Up Ollama with Open WebUI.....	25
Figure 28: Downloading Model for Inferencing.....	26
Figure 29: Selecting Model for Inferencing.....	26
Figure 30: Inferencing with Open WebUI and Ollama.....	27

## 1.0 Introduction

---

This guide is to help users install and run Ollama with Open WebUI on Intel Hardware Platform on **Windows\* 11** and **Ubuntu\* 22.04 LTS**.

Throughout this guide, you will learn to:

1. Install Intel GPU Driver and oneAPI Base Toolkit (Windows / Ubuntu).
2. Install Python\* 3.11 (Windows / Ubuntu).
3. Install IPEX-LLM for Ollama (Windows / Ubuntu).
4. Install Open WebUI (Windows / Ubuntu).
5. Configure Open WebUI with Ollama.

### 1.1 Verified Hardware

System Verified Hardware:

- CPU: Intel® Core™ i9-13900K Processor
- GPU: Intel® Arc™ A770 Graphics
- RAM: 32 GB RAM

## 2.0 Windows 11 Installation Guide

**Note:** If you are using the Ubuntu operating system, skip this section, and move to section 3.0.

### 2.1 Step 1: Installing Intel GPU Driver and oneAPI Base Toolkit (Windows)

#### 2.1.1 Installing GPU Driver

Click the [link](#) to download and install the latest Intel® Arc™ Graphics GPU Driver from the official Intel download page. A system reboot is necessary after the installation process is complete.

Figure 1: Intel® Arc™ Graphics GPU Driver Official Download Page

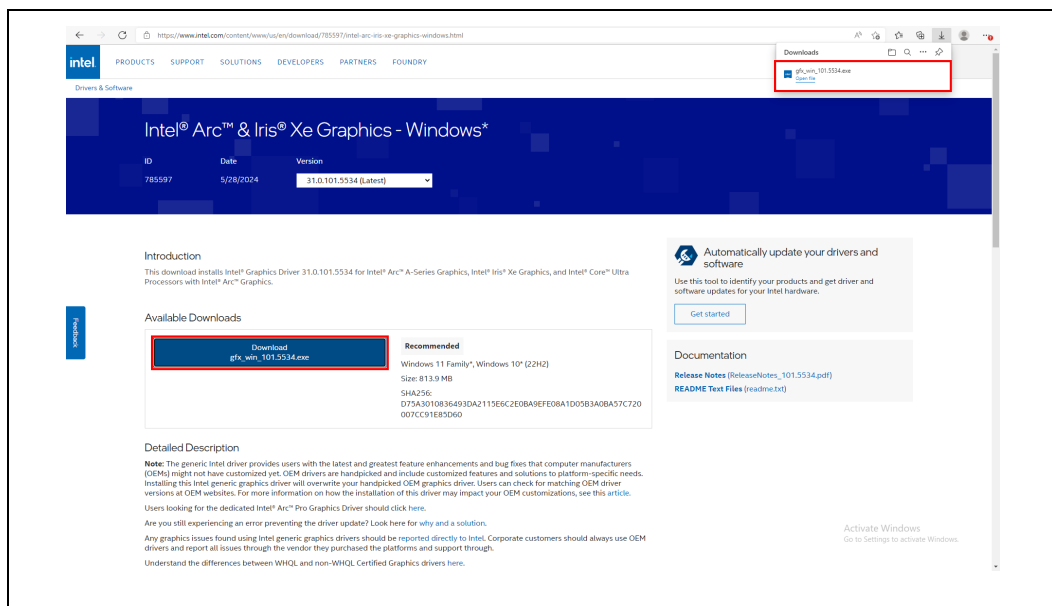
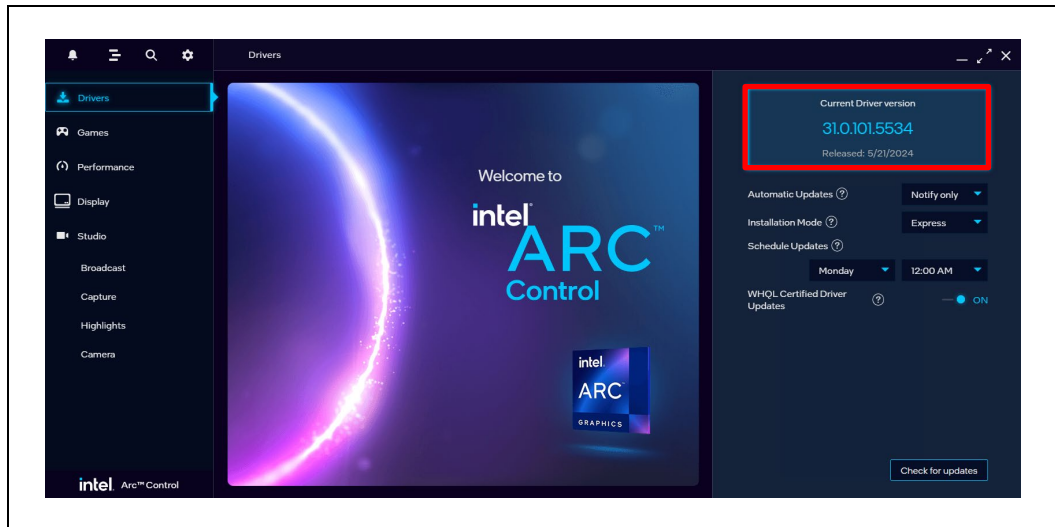


Figure 2: Intel® Arc™ Graphics GPU Driver Installation



### 2.1.1.1 Successful Installation

Figure 3: Intel® Arc™ Graphics Driver Dashboard



**Note:** Ensure that your GPU driver is updated to version 31.0.101.5333 or latest, to meet the installation requirements.

### 2.1.2 Installing Intel oneAPI Base Toolkit

To install the Intel oneAPI Base Toolkit, click the [link](#) to begin downloading and installing.

Figure 4: Intel oneAPI Base Toolkit Download Page

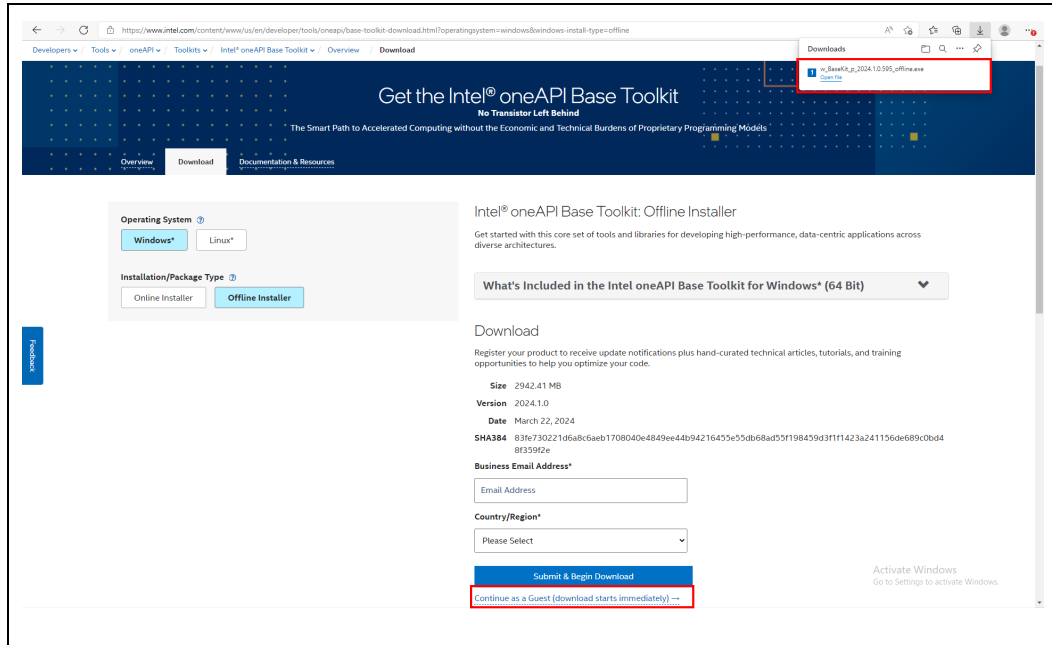


Figure 5: Installation of Intel oneAPI Base Toolkit

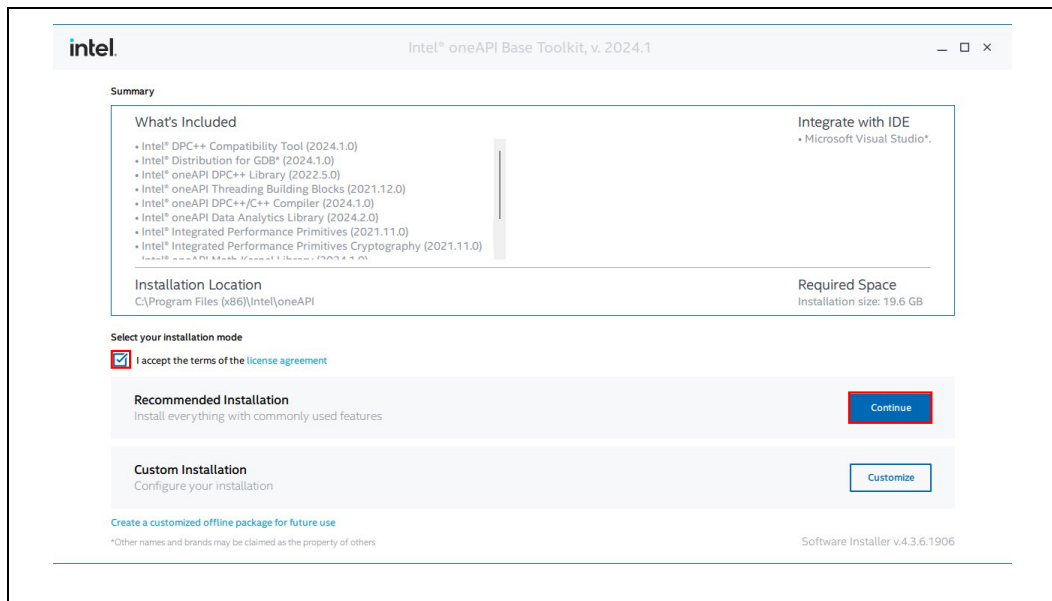




Figure 6: Installation of Intel oneAPI Base Toolkit

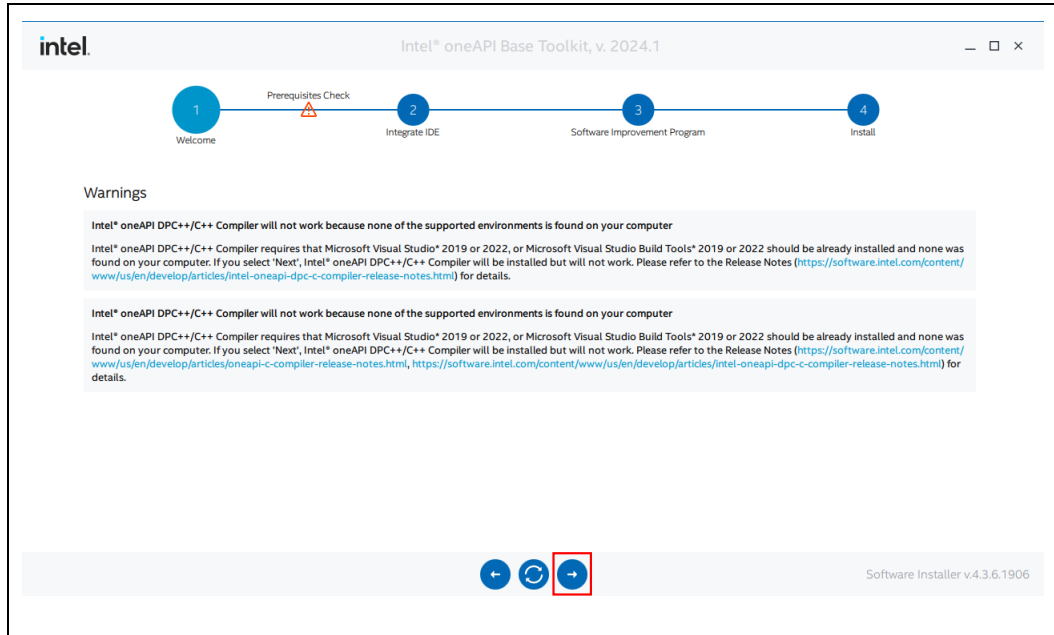
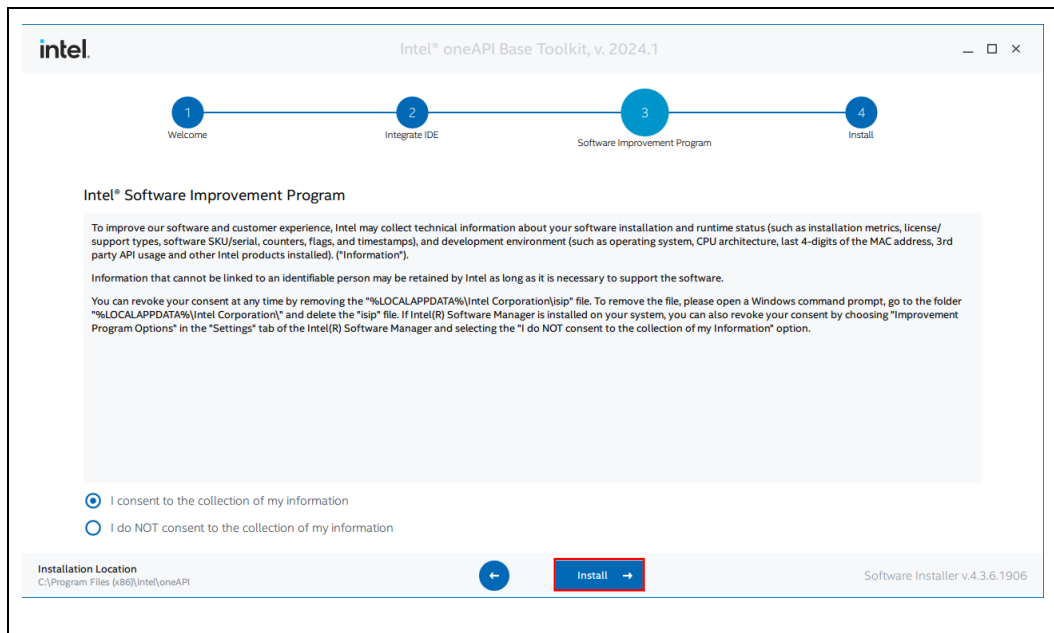
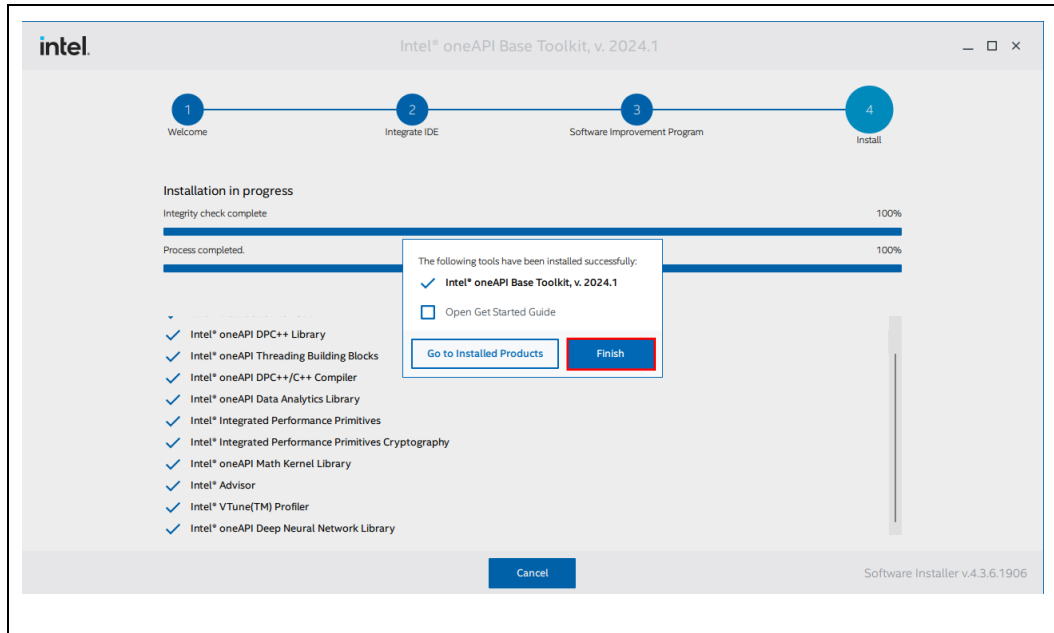


Figure 7: Installation of Intel oneAPI Base Toolkit



### 2.1.2.1 Successful Installation

Figure 8: Successful Installation of Intel oneAPI Base Toolkit



## 2.2 Step 2: Installing Python 3.11 (Windows)

To begin installing Python 3.11 for Windows, click the [link](#) to download and install Python directly from the official Python download page. Ensure that the option to add python.exe to PATH is enabled before installing.

Figure 9: Python 3.11 Official Download Page

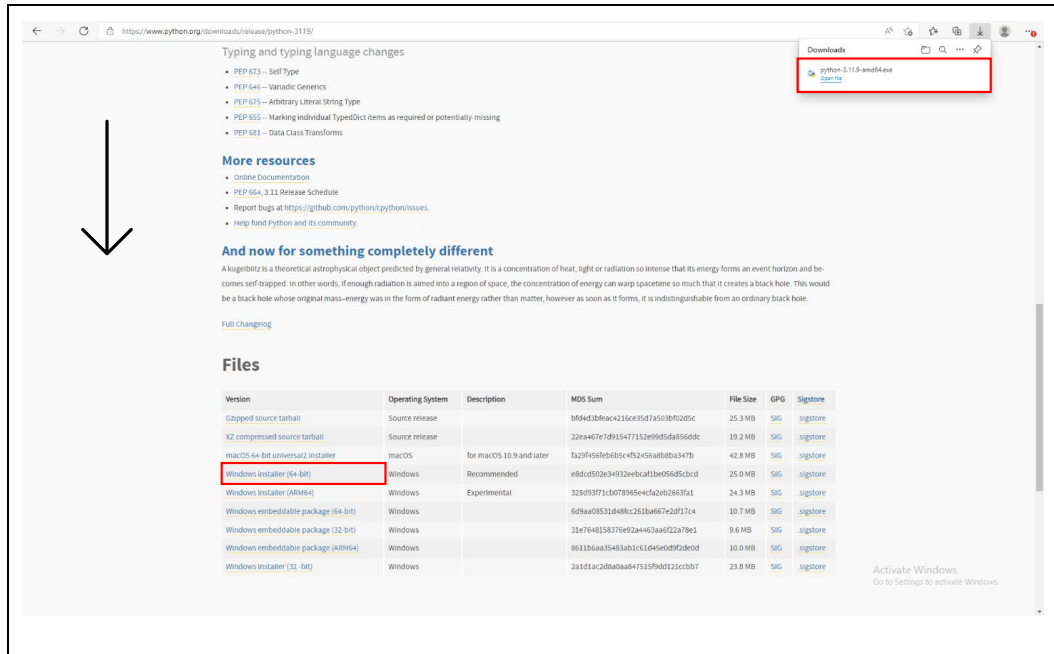


Figure 10: Python 3.11 Installation Process

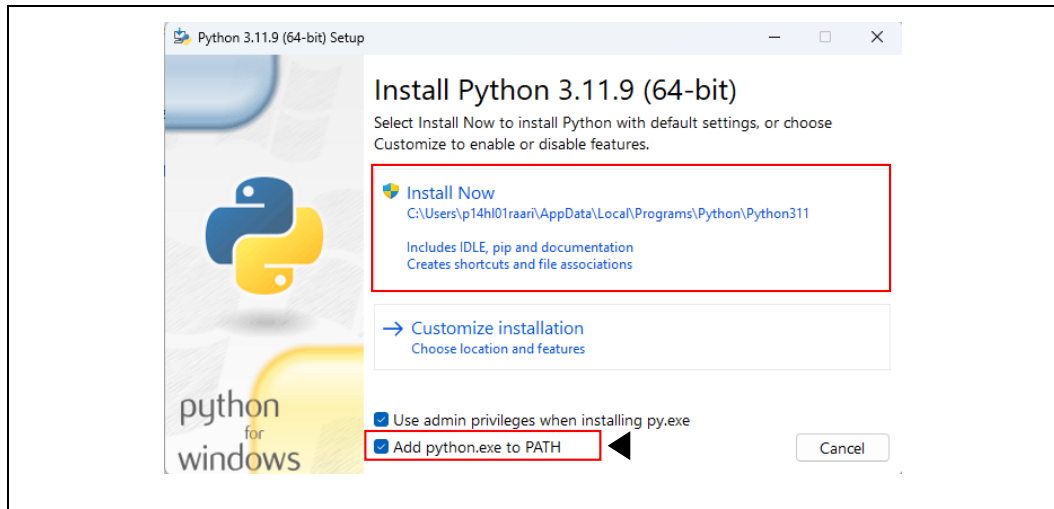


Figure 11: Python 3.11 Installation Process

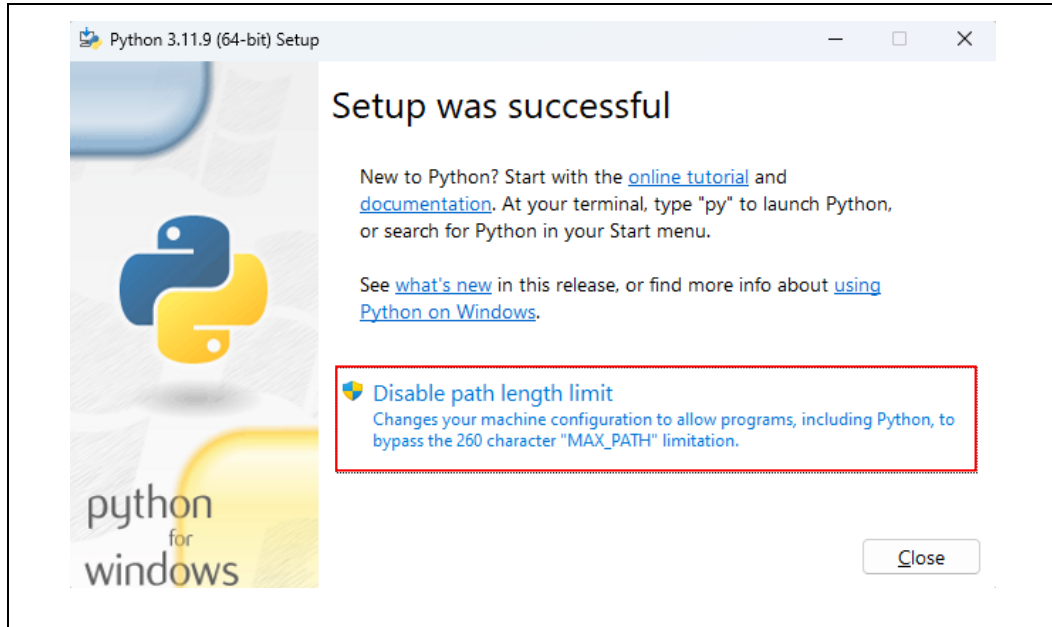
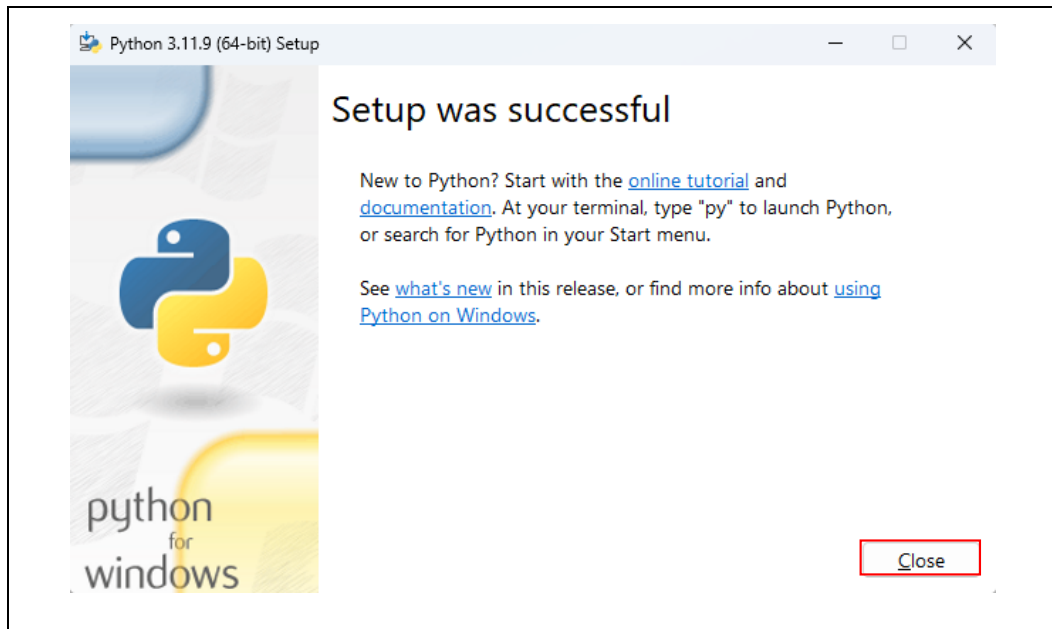


Figure 12: Python 3.11 Installation Process



### 2.2.1 Successful Installation

To ensure the successful installation of Python 3.11, follow these steps:

1. Open the Start menu and type **cmd** into the search bar.
2. Right-click on "Command Prompt" in the search results.
3. Select "Run as administrator".
4. Execute the command: **\$ python -v**

Figure 13: Steps on Opening Command Prompt as Administrator

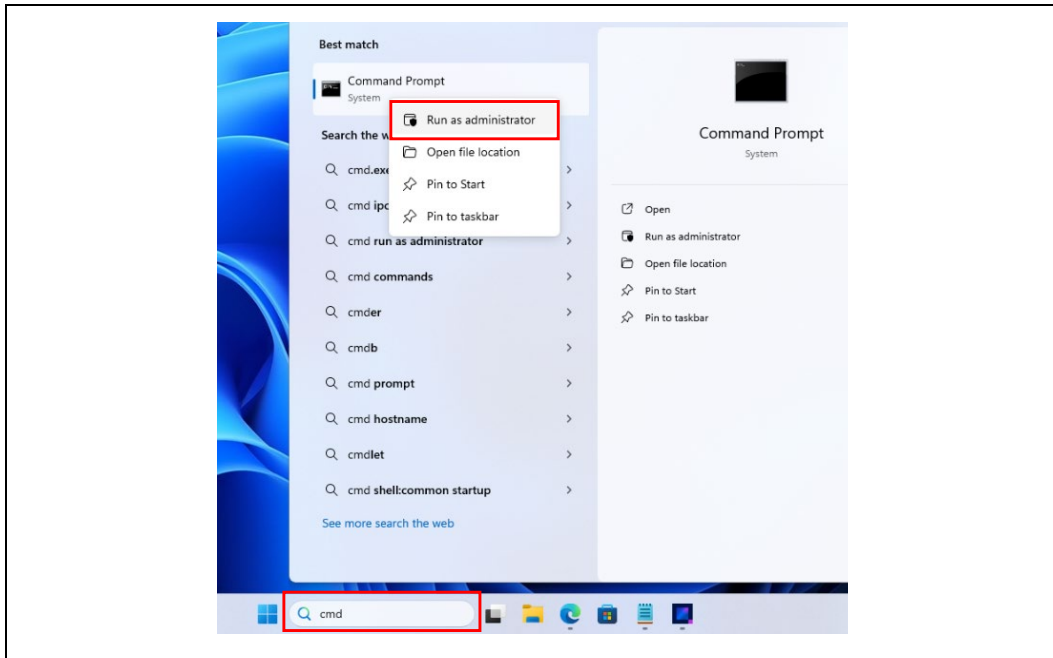
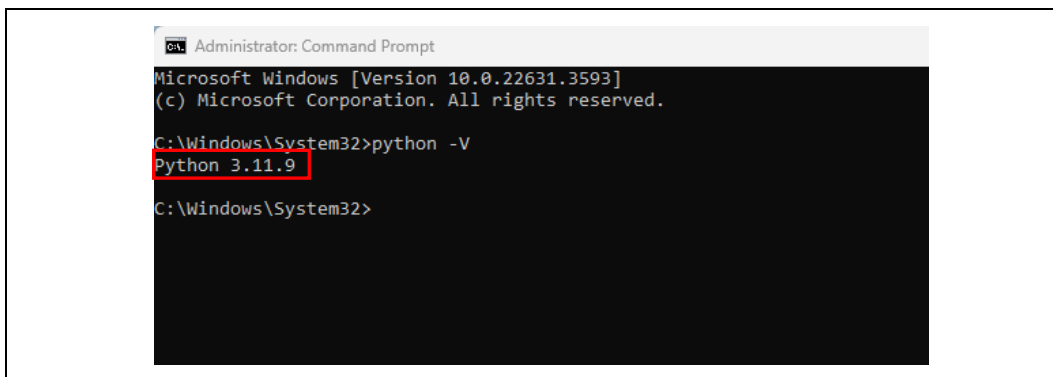


Figure 14: Command Prompt Output to Ensure Successful Installation of Python 3.11



## 2.3 Step 3: Installing IPEX-LLM for Ollama (Windows)

### 2.3.1 Installing and Initializing Ollama with Intel GPU

To use Ollama with Intel GPU, ensure that ipex-llm[cpp] is installed. To achieve this, enter the command below onto the same command prompt from before with administrator mode to install and initialize Ollama.

```
$ python -m venv llm_env
$ C:\Windows\System32\llm_env\Scripts\activate.bat
$ pip install --pre --upgrade ipex-llm[cpp]
$ mkdir llama-cpp
$ cd llama-cpp
$ init-ollama.bat
```

### 2.3.2 Run Ollama Serve with Intel GPU

You may execute the commands below to launch the Ollama service:

```
$ "C:\Program Files (x86)\Intel\oneAPI\setvars.bat"
$ set OLLAMA_NUM_GPU=999
$ set no_proxy=localhost,127.0.0.1
$ set ZES_ENABLE_SYSMAN=1
$ set SYCL_CACHE_PERSISTENT=1
$ ollama serve
```

**Note:** Set the OLLAMA\_NUM\_GPU to 999 to ensure all layers of your models are running on Intel GPU, otherwise, some layers may run on CPU.

**Note:** If you're under a network proxy, execute the following command before running ollama serve to set up network proxy for Ollama:

```
$ set HTTP_PROXY=http://proxy.com:port
$ set HTTPS_PROXY=http://proxy.com:port
```

### 2.3.3 Successful Installation

The console will display message similar to the following indicating a successful launch of the ollama service. Ensure that this terminal is not closed:

Figure 15: Successful Installation of Ollama Service

```

Administrator: Intel()oneAPI - ollama serve
time=2024-06-10T01:57:53.941-07:00 level=INFO source=images.go:736 msg="total unused blobs removed: 0"
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env: export GIN_MODE=release
- using code: gin.SetMode(gin.ReleaseMode)
[GIN-debug] POST    /api/pull          -> github.com/ollama/ollama/server.(*Server).PullModelHandler-fm (5 handlers)
[GIN-debug] POST    /api/generate     -> github.com/ollama/ollama/server.(*Server).GenerateHandler-fm (5 handlers)
[GIN-debug] POST    /api/chat         -> github.com/ollama/ollama/server.(*Server).ChatHandler-fm (5 handlers)
[GIN-debug] POST    /api/embeddings   -> github.com/ollama/ollama/server.(*Server).EmbeddingsHandler-fm (5 handlers)
[GIN-debug] POST    /api/create       -> github.com/ollama/ollama/server.(*Server).CreateModelHandler-fm (5 handlers)
[GIN-debug] POST    /api/push         -> github.com/ollama/ollama/server.(*Server).PushModelHandler-fm (5 handlers)
[GIN-debug] POST    /api/copy         -> github.com/ollama/ollama/server.(*Server).CopyModelHandler-fm (5 handlers)
[GIN-debug] DELETE  /api/delete       -> github.com/ollama/ollama/server.(*Server).DeleteModelHandler-fm (5 handlers)
[GIN-debug] POST    /api/show         -> github.com/ollama/ollama/server.(*Server).ShowModelHandler-fm (5 handlers)
[GIN-debug] POST    /api/blobs/:digest -> github.com/ollama/ollama/server.(*Server).CreateBlobHandler-fm (5 handlers)
[GIN-debug] HEAD    /api/blobs/:digest -> github.com/ollama/ollama/server.(*Server).HeadBlobHandler-fm (5 handlers)
[GIN-debug] GET    /api/ps          -> github.com/ollama/ollama/server.(*Server).ProcessHandler-fm (5 handlers)
[GIN-debug] POST   /v1/chat/completions -> github.com/ollama/ollama/server.(*Server).ChatHandler-fm (6 handlers)
[GIN-debug] GET    /                -> github.com/ollama/ollama/server.(*Server).ListModelHandler-fm (5 handlers)
[GIN-debug] GET    /api/tags        -> github.com/ollama/ollama/server.(*Server).ListModelHandler-fm (5 handlers)
[GIN-debug] GET    /api/version     -> github.com/ollama/ollama/server.(*Server).GenerateRoutes.func2 (5 handlers)
[GIN-debug] HEAD    /                -> github.com/ollama/ollama/server.(*Server).GenerateRoutes.func1 (5 handlers)
[GIN-debug] HEAD    /api/tags        -> github.com/ollama/ollama/server.(*Server).ListModelHandler-fm (5 handlers)
[GIN-debug] HEAD    /api/version     -> github.com/ollama/ollama/server.(*Server).GenerateRoutes.func2 (5 handlers)
time=2024-06-10T01:57:53.945-07:00 level=INFO source=routes.go:1074 msg="Listening on 127.0.0.1:11434 (version 0.0.0)"
time=2024-06-10T01:57:53.945-07:00 level=INFO source=payload.go:44 msg="Dynamic LHM libraries [cpu cpu_avx cpu_avx2]"
time=2024-06-10T01:57:53.953-07:00 level=INFO source=types.go:71 msg="inference compute" id=0 library=cpu compute="" driver=0.0 name="" total="31.6 GiB" available="23.3 GiB"

```

## 2.4 Step 4: Installing Open WebUI (Windows)

### 2.4.1 Install Open WebUI

Open a new command prompt window with **administrator mode** and run the following command:

```
$ C:\Windows\System32\llm_env\Scripts\activate.bat
$ pip install open-webui==0.2.5
```

### 2.4.2 Start Open WebUI Server

Once installed, you may start the server using the following command:

```
$ open-webui serve
```

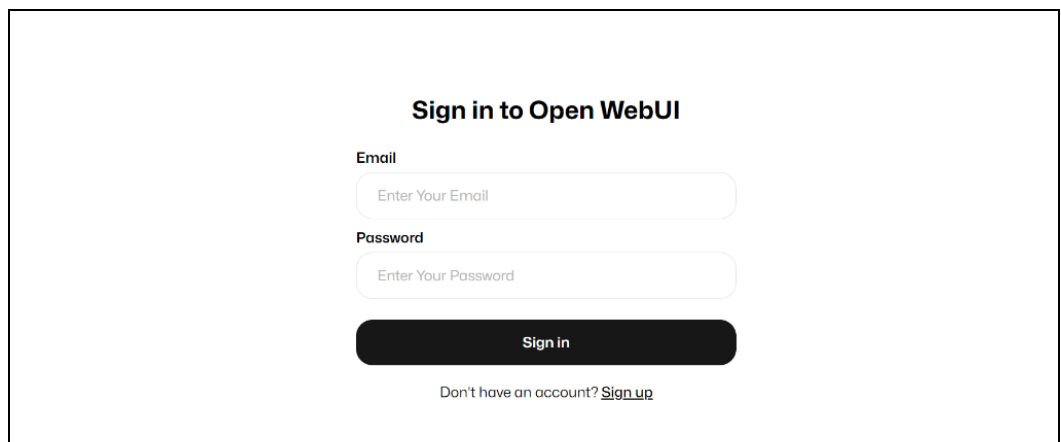
**Note:** If you're under a network proxy, execute the following command before running `open-webui serve` to set up network proxy for Open WebUI:

```
$ set http_proxy=http://proxy.com:port
$ set https_proxy=http://proxy.com:port
$ set no_proxy=localhost,127.0.0.1
```

### 2.4.3 Successful Installation

In this step, you may access the Open WebUI using <http://localhost:8080> once the installation is successful.

Figure 16: Successful Installation of Open WebUI



The screenshot shows a web browser window displaying the "Sign in to Open WebUI" page. The page has a white background with a black border. At the top center, the text "Sign in to Open WebUI" is displayed in a bold, black font. Below this, there are two input fields: "Email" and "Password". The "Email" field has a placeholder text "Enter Your Email" and the "Password" field has a placeholder text "Enter Your Password". Below these fields is a black button with the text "Sign in" in white. At the bottom center, there is a link that says "Don't have an account? [Sign up](#)".



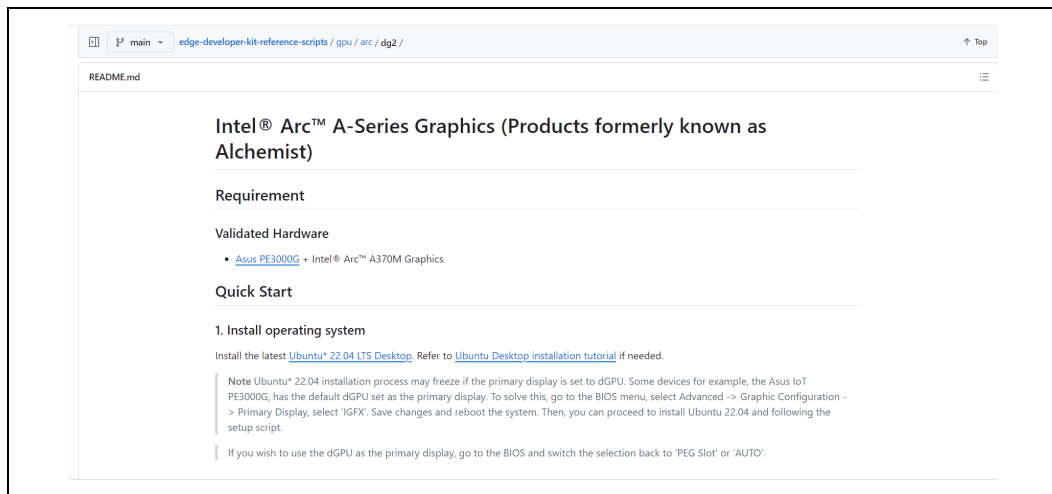
## 3.0 Ubuntu Installation Guide

### 3.1 Step 1: Installing Intel GPU Driver and oneAPI Base Toolkit (Ubuntu)

#### 3.1.1 Installing GPU Driver

To start, click the [link](#) and follow the steps in the README.md to begin installing the Intel® Arc™ Graphics GPU Driver.

Figure 17: GPU Driver Installation Page



#### 3.1.2 Installing Intel oneAPI Base Toolkit

To install the Intel oneAPI Base Toolkit, click the [link](#) and follow the steps provided on the webpage.

Figure 18: Intel oneAPI Base Toolkit Download Page

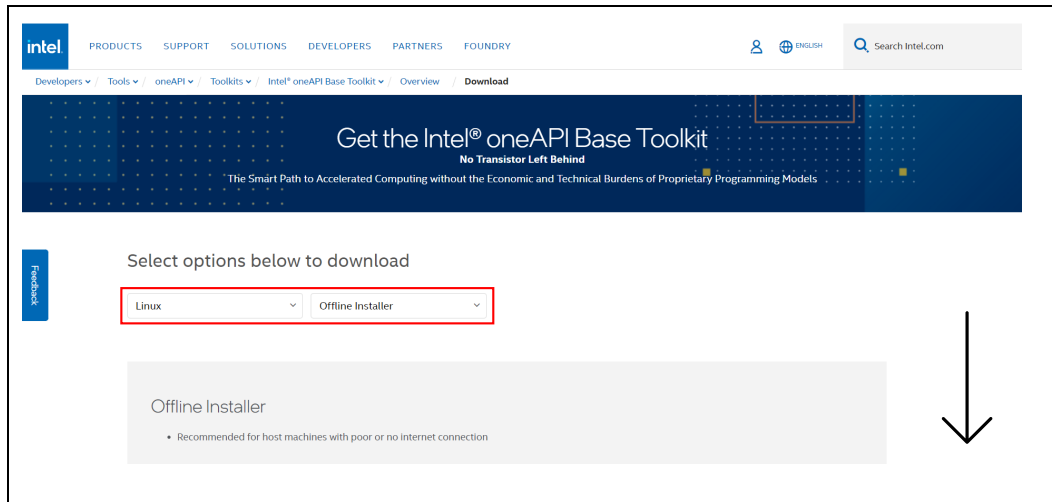


Figure 19: Intel oneAPI Base Toolkit Command Line Guide

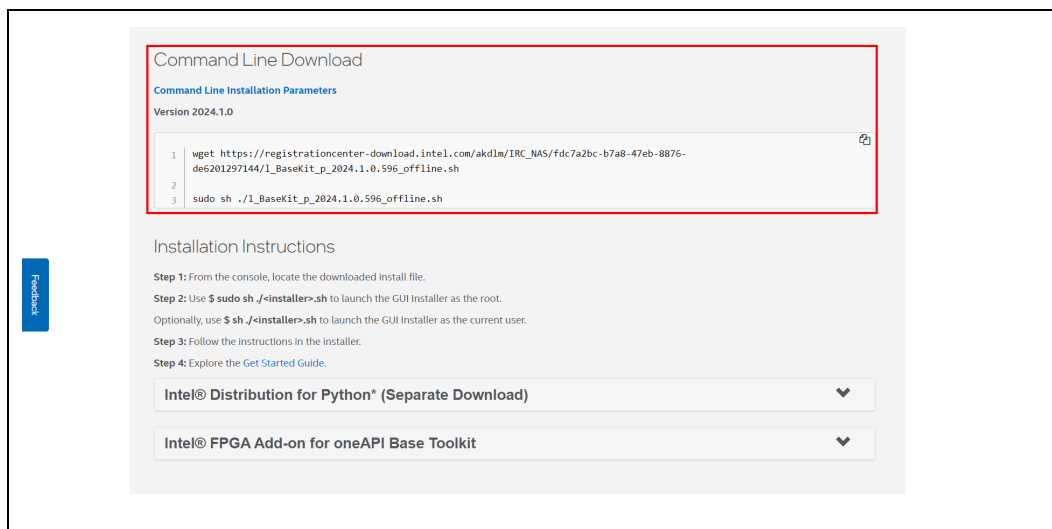


Figure 20: Installation of Intel oneAPI Base Toolkit

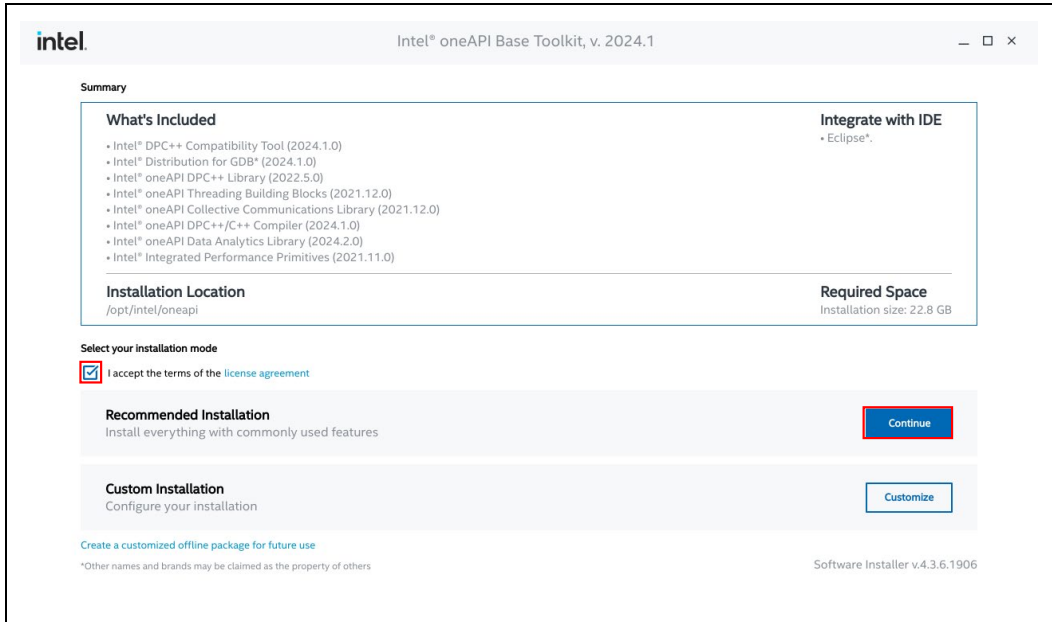
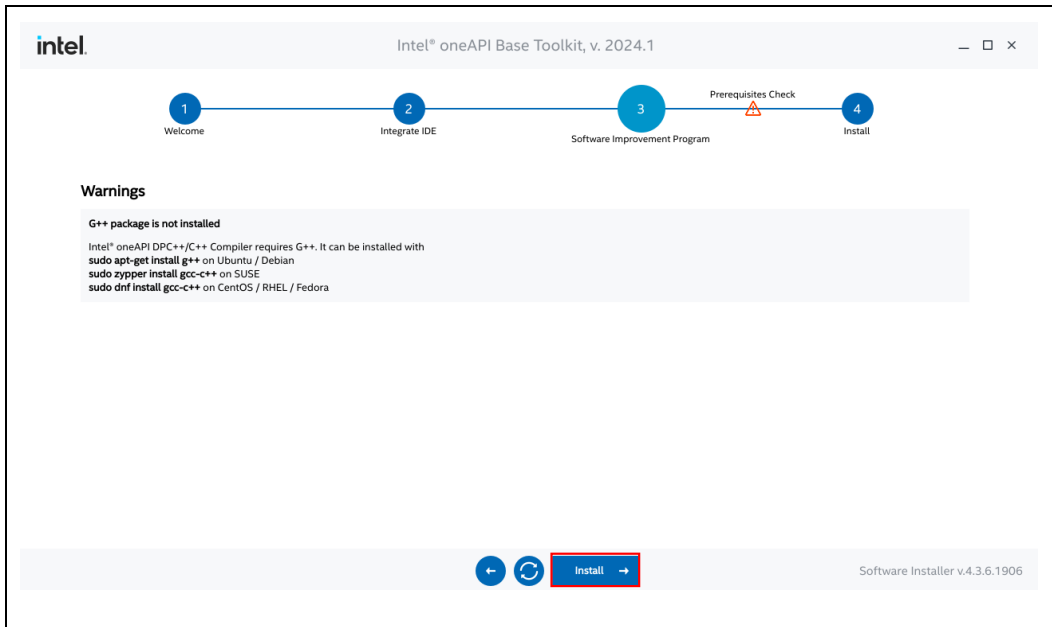
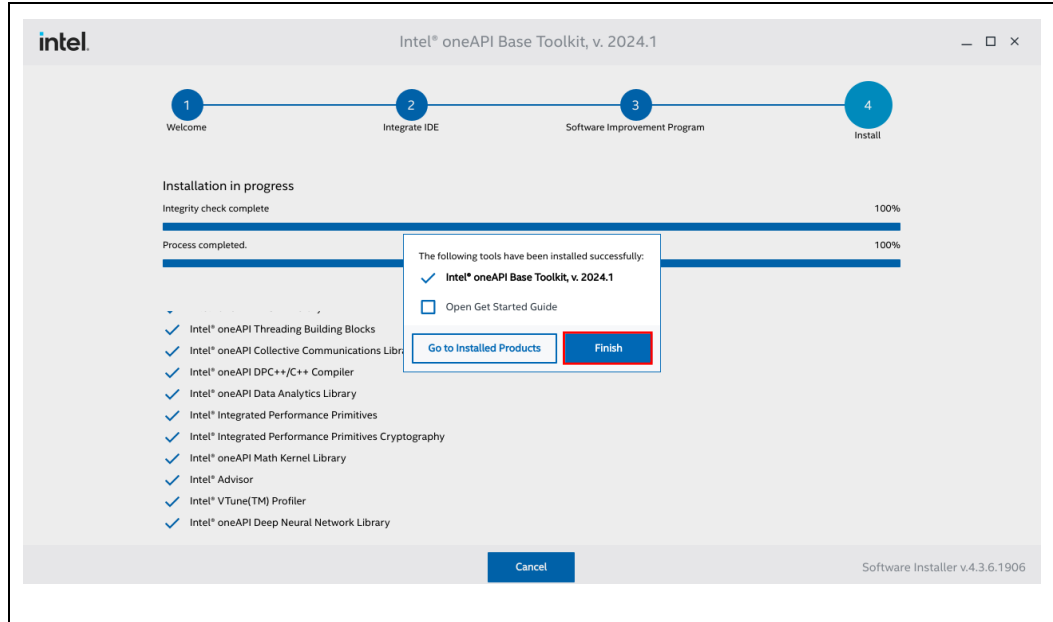


Figure 21: Installation of Intel oneAPI Base Toolkit



### 3.1.2.1 Successful Installation

Figure 22: Successful Installation of Intel oneAPI Base Toolkit



## 3.2 Step 2: Installing Python 3.11 (Ubuntu)

### 3.2.1 Update System Packages

Next, update the system packages in the Ubuntu 22.04 terminal using:

```
$ sudo apt update
$ sudo apt upgrade
```

### 3.2.2 Adding PPA Repository

Add the deadsnakes PPA repository to the system:

```
$ sudo add-apt-repository ppa:deadsnakes/ppa
```

### 3.2.3 Python 3.11 Installation

Use this command to install Python 3.11 and required dependencies on Ubuntu 22.04:

```
$ sudo apt install python3.11 -y
$ sudo apt install python3.11-venv -y
```

### 3.2.4 Validate Python Version

Use this command to validate Python 3.11 installation on Ubuntu 22.04:

```
$ python3.11 -V
```

**Output:**

```
Python 3.11.xx
```

## 3.3 Step 3: Installing IPEX-LLM for Ollama (Ubuntu)

### 3.3.1 Installing and initializing Ollama with Intel GPU

To use Ollama with Intel GPU, ensure that `ipex-llm[cpp]` is installed. To achieve this, enter the command below onto the terminal to install and initialize Ollama.

```
$ python3.11 -m venv llm_env
$ source llm_env/bin/activate
$ pip install --pre --upgrade ipex-llm[cpp]
$ mkdir llama-cpp
$ cd llama-cpp
$ init-ollama
```

### 3.3.2 Run Ollama Serve with Intel GPU

You may execute the commands below to launch the Ollama service:

```
$ export OLLAMA_NUM_GPU=999
$ export no_proxy=localhost,127.0.0.1
$ export ZES_ENABLE_SYSMAN=1
$ source /opt/intel/oneapi/setvars.sh
$ export SYCL_CACHE_PERSISTENT=1
$ ./ollama serve
```

**Note:** Set the `OLLAMA_NUM_GPU` to **999** to ensure all layers of your models are running on Intel GPU, otherwise, some layers may run on CPU.

**Note:** To allow the service to accept connections from all IP addresses, use:

```
$ OLLAMA_HOST=0.0.0.0 ./ollama serve
```

**Note:** If your local LLM is running on Intel® Arc™ A-Series Graphics with Linux OS (Kernel 6.2), it is recommended to additionally add the following environment command for optimal performance before executing `./ollama serve`:

```
$ export SYCL_PI_LEVEL_ZERO_USE_IMMEDIATE_COMMANDLISTS=1
```

### 3.3.3 Successful Installation

The console will display message similar to the following indicating a successful launch of the ollama service. Ensure that this terminal is not closed:

Figure 23: Successful Installation of Ollama Service

```
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env:   export GIN_MODE=release
- using code: gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST   /api/pull          --> github.com/ollama/ollama/server.(*Server).PullModelHandler-fm (5 handlers)
[GIN-debug] POST   /api/generate     --> github.com/ollama/ollama/server.(*Server).GenerateHandler-fm (5 handlers)
[GIN-debug] POST   /api/chat         --> github.com/ollama/ollama/server.(*Server).ChatHandler-fm (5 handlers)
[GIN-debug] POST   /api/embeddings   --> github.com/ollama/ollama/server.(*Server).EmbeddingsHandler-fm (5 handlers)
[GIN-debug] POST   /api/create       --> github.com/ollama/ollama/server.(*Server).CreateModelHandler-fm (5 handlers)
[GIN-debug] POST   /api/push        --> github.com/ollama/ollama/server.(*Server).PushModelHandler-fm (5 handlers)
[GIN-debug] POST   /api/copy        --> github.com/ollama/ollama/server.(*Server).CopyModelHandler-fm (5 handlers)
[GIN-debug] DELETE  /api/delete       --> github.com/ollama/ollama/server.(*Server).DeleteModelHandler-fm (5 handlers)
[GIN-debug] POST   /api/show        --> github.com/ollama/ollama/server.(*Server).ShowModelHandler-fm (5 handlers)
[GIN-debug] POST   /api/blobs/:digest --> github.com/ollama/ollama/server.(*Server).CreateBlobHandler-fm (5 handlers)
[GIN-debug] HEAD   /api/blobs/:digest --> github.com/ollama/ollama/server.(*Server).HeadBlobHandler-fm (5 handlers)
[GIN-debug] GET    /api/ps          --> github.com/ollama/ollama/server.(*Server).ProcessHandler-fm (5 handlers)
[GIN-debug] POST   /v1/chat/completions --> github.com/ollama/ollama/server.(*Server).ChatHandler-fm (0 handlers)
[GIN-debug] GET    /                --> github.com/ollama/ollama/server.(*Server).GenerateRoutes.func1 (5 handlers)
[GIN-debug] GET    /api/tags        --> github.com/ollama/ollama/server.(*Server).ListModelHandler-fm (5 handlers)
[GIN-debug] GET    /api/versions   --> github.com/ollama/ollama/server.(*Server).GenerateRoutes.func2 (5 handlers)
[GIN-debug] HEAD   /                --> github.com/ollama/ollama/server.(*Server).GenerateRoutes.func1 (5 handlers)
[GIN-debug] HEAD   /api/tags       --> github.com/ollama/ollama/server.(*Server).ListModelHandler-fm (5 handlers)
[GIN-debug] HEAD   /api/versions   --> github.com/ollama/ollama/server.(*Server).GenerateRoutes.func2 (5 handlers)
time=2024-05-29T11:25:32.886+08:00 level=INFO source=routes.go:1074 msg="Listening on 127.0.0.1:11434 (version 0.8.0)"
time=2024-05-29T11:25:32.886+08:00 level=INFO source=payload.go:30 msg="extracting embedded files" dir=/tmp/ollama2885753/runners
time=2024-05-29T11:25:32.849+08:00 level=INFO source=payload.go:44 msg="dynamic LLN libraries [cpu avx2 cpu_avx]"
time=2024-05-29T11:25:32.853+08:00 level=INFO source=types.go:71 msg="inference compute" id=0 library=cpu compute="" driver=0.0 name="" total="30.9 GiB" available="3.1 GiB"
```

## 3.4 Step 4: Installing Open WebUI (Ubuntu)

### 3.4.1 Install Open WebUI

Open a new terminal using **CTRL + ALT + T** and run the following command:

```
$ source llm_env/bin/activate
$ pip install open-webui==0.2.5
```

### 3.4.2 Start Open WebUI Server

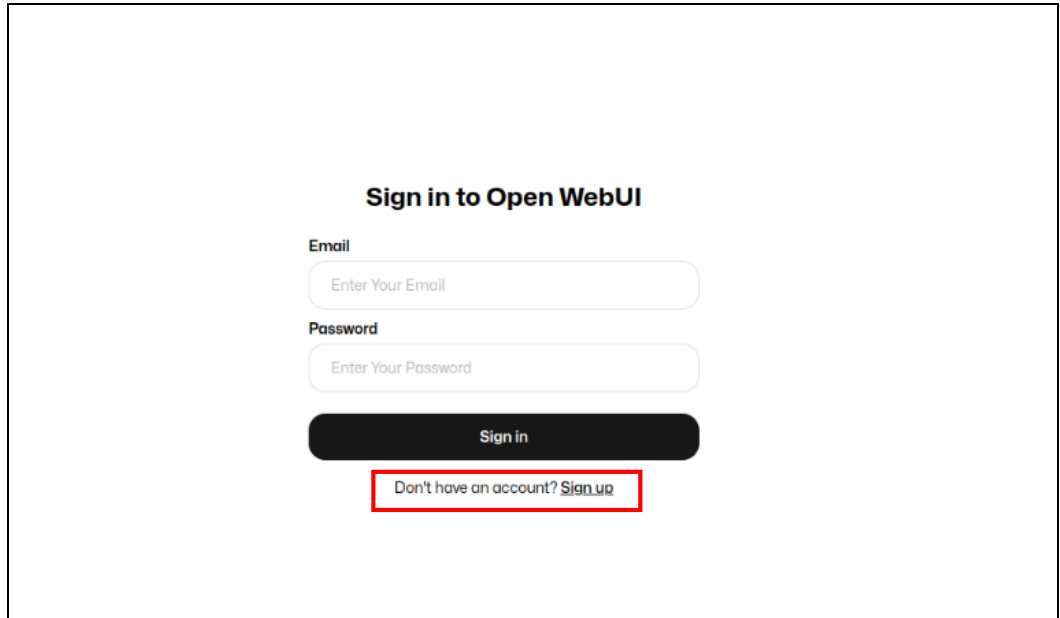
Once installed, you may start the server using the following command:

```
$ open-webui serve
```

### 3.4.3 Successful Installation

In this step, you may access the Open WebUI using <http://localhost:8080> once the installation is successful.

Figure 24: Successful Installation of Open WebUI



## 4.0 *Configuring Open WebUI with Ollama*

---

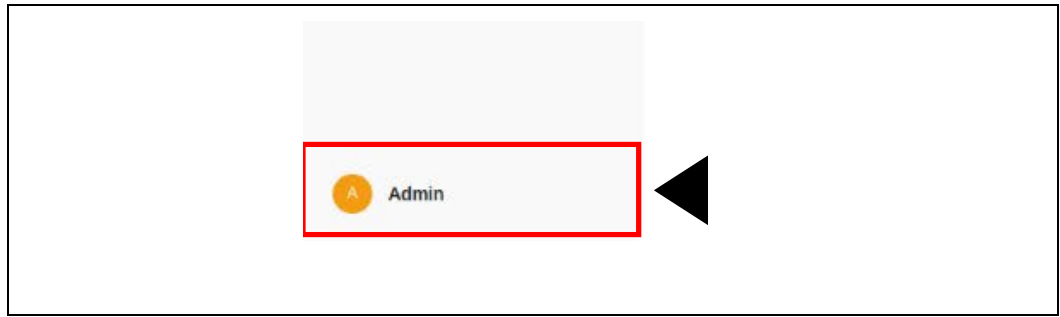
### 4.1 Step 5: Configuring Open WebUI with Ollama

#### 4.1.1 Setting Up Ollama with Open WebUI

To set up Ollama with Open WebUI, once logged in, follow the steps below:

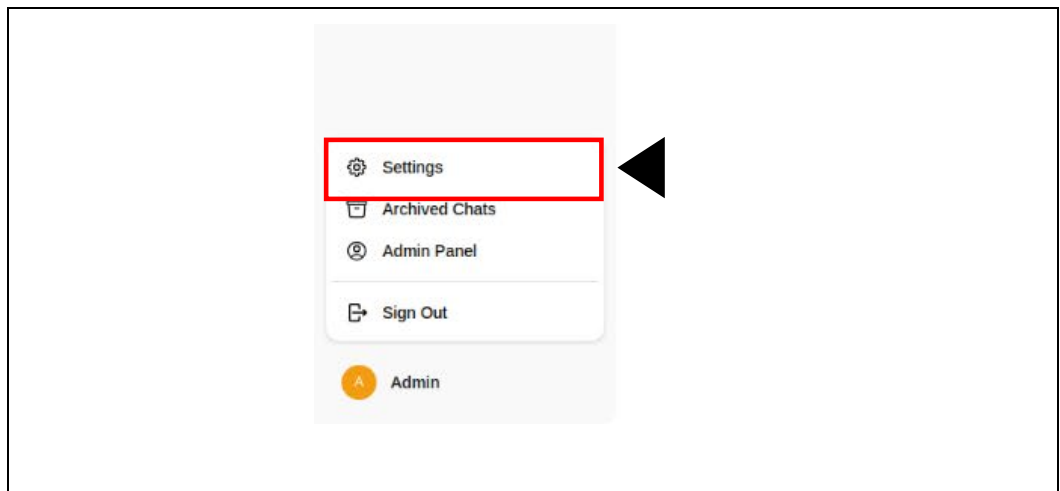
1. Click on your profile at the bottom left of the page.

Figure 25: Step 1 of Setting Up Ollama with Open WebUI



2. Navigate to Settings

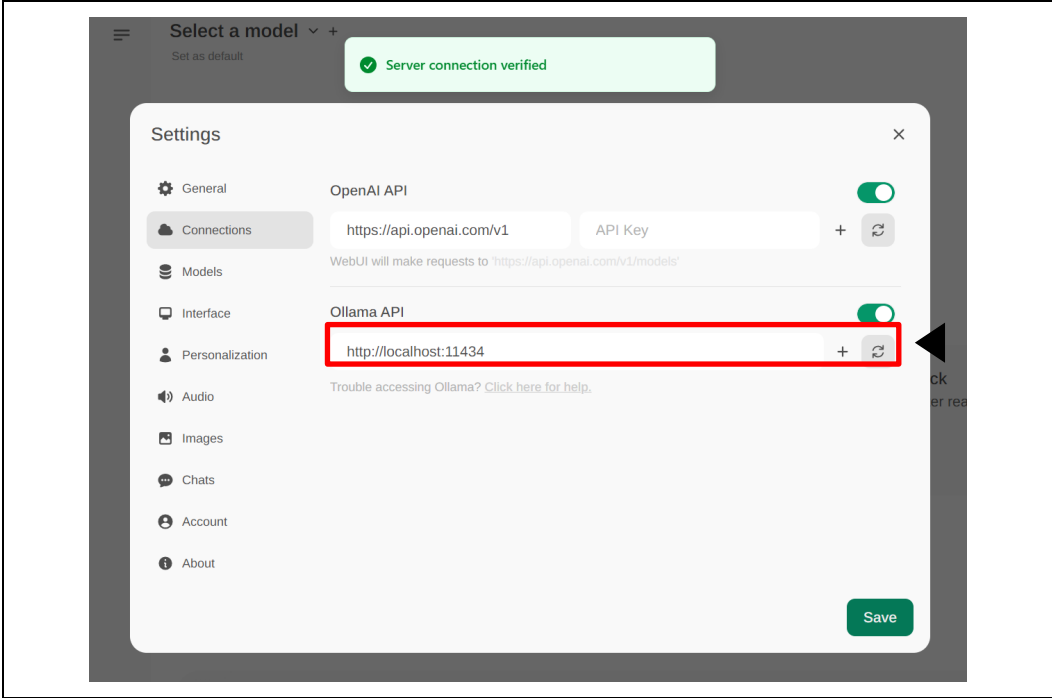
Figure 26: Step 2 of Setting Up Ollama with Open WebUI





3. Go to Connections to link the Open WebUI with your Ollama service

Figure 27: Step 3 of Setting Up Ollama with Open WebUI



## 4.2 Download Model and Inferencing

To download your desired model, you may navigate to **settings > models** to pull your model and start to perform inferences once the model has completed downloading:

Figure 28: Downloading Model for Inferencing

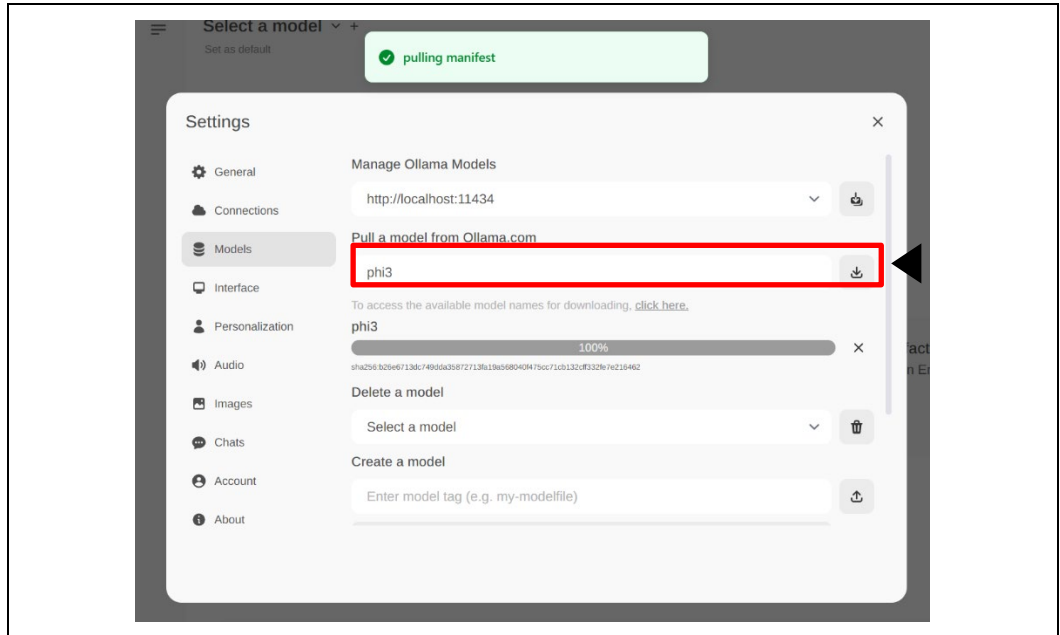


Figure 29: Selecting Model for Inferencing

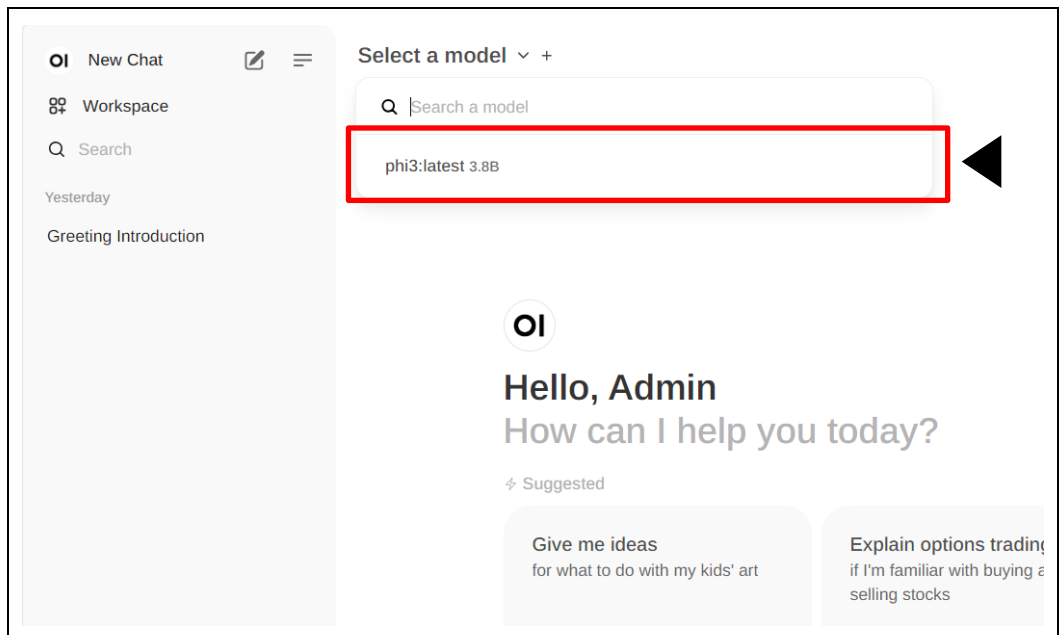
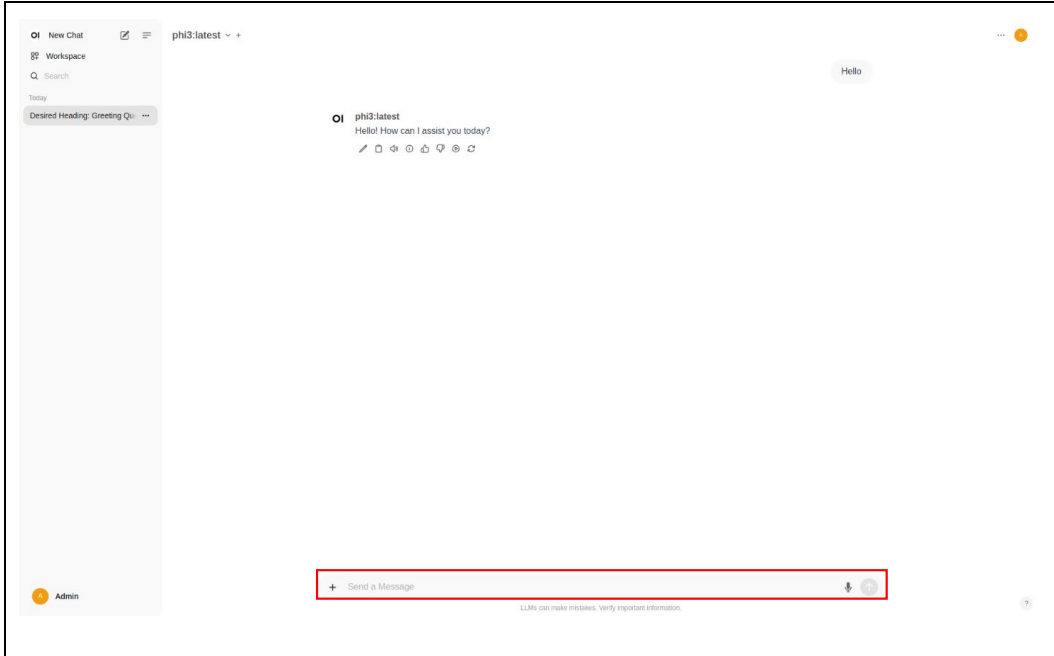


Figure 30: Inferencing with Open WebUI and Ollama



§

## Revision History

---

Date	Revision	Description
June 2024	1.0	Initial release.