

How to Install Open WebUI - An Opensource Web Interface for Running LLMs

Learn how to install Open WebUI, a user-friendly open-source interface for running Large Language Models (LLMs) on your system with step-by-step instructions.

Contents

| | | |
|-----|--|----|
| 01 | Introduction | 3 |
| 02 | Prerequisites | 3 |
| 03 | Install Ollama | 3 |
| 04 | Install Open WebUI | 6 |
| 05 | Option 1: Install Open WebUI Using Pip | 6 |
| 06 | Option 2: Install Open WebUI Using Docker | 9 |
| 07 | Configure Nginx as a Reverse Proxy to Run Open WebUI | 11 |
| 08 | Access Open WebUI | 15 |
| 09 | Install and Run LLMs in Open WebUI | 17 |
| 010 | Conclusion | 19 |

Introduction

Open WebUI is a self-hosted, feature-rich, and user-friendly open-source web interface that lets you run large language models (LLMs) using Ollama and OpenAI-compatible APIs. You can extend Open WebUI with AI models and plugins to suit your needs.

This article explains how to install Open WebUI and run large language models (LLMs) using Ollama on a Vultr Cloud GPU instance. You will also configure Open WebUI with secure SSL certificates to expose the web interface using a custom domain.

Prerequisites

Before you begin:

- Deploy a [Vultr Cloud GPU instance](#) with at least 8 GB RAM to run LLMs. Choose:
 - 8 GB for 7B models
 - 16 GB for 13B models
 - 32 GB for 33B models.
- Set up a [domain A recording pointing to the instance's public IP address](#). For example, `openwebui.example.com`.
- [Access the instance using SSH](#) as a non-root sudo user.

Install Ollama

Ollama is a lightweight, extensible framework for running open-source LLMs such as Llama, Code Llama, Mistral, and Gemma. Ollama integrates with application frameworks using APIs to create, manage, and customize models.

Open WebUI requires Ollama to run and manage LLMs. Follow the steps below to install Ollama and download a sample model to run using Open WebUI.

1. Download the Ollama installation script.

CONSOLE

```
$ wget https://ollama.ai/install.sh
```

2. Grant execute permissions to the script.

CONSOLE

```
$ sudo chmod +x install.sh
```

3. Run the script to install Ollama.

CONSOLE

```
$ sudo ./install.sh
```

Output:

```
>>> Installing ollama to /usr/local
>>> Downloading Linux amd64 bundle
#####
##### 100.0%>>> Creating ollama user...
>>> Adding ollama user to render group...
>>> Adding ollama user to video group...
>>> Adding current user to ollama group...
>>> Creating ollama systemd service...
>>> Enabling and starting ollama service...
Created symlink /etc/systemd/system/default.target.wants/ollama.service → /etc/systemd/system/ollama.service.
>>> NVIDIA GPU installed.
```

4. Enable the Ollama system service to start at boot.

CONSOLE

CONSOLE

```
$ sudo python3.11 -m pip install -U Pillow pyopenssl
```

4. Run Open WebUI and verify it does not return errors.

CONSOLE

```
$ sudo open-webui serve
```

Output:

```
 / _ \ _ _ _ _ _ _ _ _ \ \ / / _ | | | | | | | | | |
| | | | ' \ / _ \ ' \ \ \ \ / / _ \ ' \ | | | | | |
| | | | ) | _ / | | | \ \ \ / / _ / ) | | | | | |
 \ \ / | . _ / \ | | | | \ \ \ / \ | . _ / \ \ / | |
  | |
```

v0.4.6 - building the best open-source AI user interface.

<https://github.com/open-webui/open-webui>

```
INFO: Started server process [16779]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
```

Press Ctrl + C to stop the application.

Create a System Service to Run Open WebUI

If you install Open WebUI using Pip, the application runs using the `open-webui serve` command. You should create a system service to manage the Open WebUI processes without running the `open-webui serve` command directly. Follow the steps below.

1. Create a new `/usr/lib/systemd/system/openwebui.service` system service file using a text editor like `vim`.

CONSOLE

```
$ sudo vim /usr/lib/systemd/system/openwebui.service
```

2. Add the following service configurations to the `/usr/lib/systemd/system/openwebui.service` file.

INI

```
[Unit]
Description=Open WebUI Service
After=network.target

[Service]
Type=simple
ExecStart=open-webui serve
ExecStop=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
```

Save and close the file.

In the above system service configuration, starting the Open WebUI service automatically runs the `open-webui serve` command on your server.

3. Reload systemd to apply the new service configuration.

CONSOLE

```
$ sudo systemctl daemon-reload
```

4. Enable the Open WebUI system service to start at boot.

CONSOLE

```
$ sudo systemctl enable openwebui.service
```

5. Start the Open WebUI service.

CONSOLE

```
$ sudo systemctl start openwebui
```

6. Test the Open WebUI service status and confirm it's running.

CONSOLE

```
$ sudo systemctl status openwebui
```

Output:

```
● openwebui.service - Open WebUI Service
   Loaded: loaded (/lib/systemd/system/openwebui.service; enabled; vendor
  preset: enabled)
   Active: active (running) since Wed 2024-12-04 20:20:29 UTC; 6s ago
 Main PID: 3007 (open-webui)
    Tasks: 9 (limit: 14212)
   Memory: 522.7M
      CPU: 6.456s
   CGroup: /system.slice/openwebui.service
           └─3007 /usr/bin/python3.11 /usr/local/bin/ open-webui serve
```

Option 2: Install Open WebUI Using Docker

1. Verify that Docker is running on your server.

CONSOLE

```
$ sudo docker ps
```

Output:

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|--------------|-------|---------|---------|--------|-------|-------|
|--------------|-------|---------|---------|--------|-------|-------|

Run `sudo systemctl start docker` to start Docker in case you receive the following error.

```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
```

2. Verify that a GPU device is available on your server.

CONSOLE

```
$ nvidia-smi
```

3. Pull the Open WebUI Docker image for Nvidia GPUs to your server.

CONSOLE

```
$ sudo docker pull ghcr.io/open-webui/open-webui:cuda
```

To use Open WebUI on a non-GPU server, download the main Docker image instead.

CONSOLE

```
$ sudo docker pull ghcr.io/open-webui/open-webui:main
```

4. Run Open WebUI and use all Nvidia GPU devices on the host server.

CONSOLE

```
$ sudo docker run -d -p 8080:8080 --gpus all --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:cuda
```

In the above command, Docker runs Open WebUI using the `open-webui:cuda` container image you downloaded earlier with the following options:

- `-p 8080:8080`: Maps the host port `8080` to the Open WebUI container port `8080` for access on your server.
- `--gpus all`: Enables Open WebUI to access all GPUs available on your server.

- `--add-host=host.docker.internal:host-gateway`: Enables the Open WebUI container to communicate with services outside Docker that are installed on the server.
- `-v open-webui:/app/backend/data`: Creates a new `open-webui` volume and mounts it to `/app/backend/data` inside the Open WebUI container.
- `--name open-webui-new`: Sets the Open WebUI container name for visibility and management.
- `--restart always`: Enables the Open WebUI container to automatically restart in case the container stops unexpectedly.
- `ghcr.io/open-webui/open-webui:cuda`: Specifies the Open WebUI docker image to run.

5. Ensure that the Open WebUI Docker container is running.

CONSOLE

```
$ sudo docker ps
```

Output:

```
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS          NAMES
d7957cd20569   ghcr.io/open-webui/open-webui:cuda  "bash start.sh"        12 seconds ago   Up 12 seconds (health: starting)  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   open-webui
```

Configure Nginx as a Reverse Proxy to Run Open WebUI

You can access the Open WebUI on the default port `8080` using your server's IP address or linked domain. Exposing the Open WebUI port in a production environment is insecure. You can use Nginx as a reverse proxy to handle secure connections to the Open WebUI's backend port using HTTP or HTTPS. In the

following steps, install Nginx and create a new virtual host configuration to access Open WebUI on your server.

1. Install Nginx.

CONSOLE

```
$ sudo apt install nginx -y
```

2. Start the Nginx system service.

CONSOLE

```
$ sudo systemctl start nginx
```

If you receive an error, stop Apache or any other application using the HTTP port `80` on your server to enable Nginx to run.

3. Create a new `/etc/nginx/sites-available/openwebui.conf` virtual host for Open WebUI.

CONSOLE

```
$ sudo vim /etc/nginx/sites-available/openwebui.conf
```

4. Add the following Nginx configurations to the `/etc/nginx/sites-available/openwebui.conf` file.

NGINX

```
server {  
    listen 80;  
    server_name openwebui.example.com;  
  
    access_log /var/log/nginx/openwebui_access.log;  
    error_log /var/log/nginx/openwebui_error.log;  
  
    location / {  
        proxy_pass http://127.0.0.1:8080;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
    }  
}
```

```
    proxy_set_header X-Forwarded-For  
$proxy_add_x_forwarded_for;  
    proxy_set_header X-Forwarded-Proto $scheme;  
  }  
}
```

Save the file.

5. Link the `openwebui.conf` configuration to the `sites-enabled` directory to enable it.

CONSOLE

```
$ sudo ln -s /etc/nginx/sites-available/openwebui.conf /etc/  
nginx/sites-enabled/openwebui.conf
```

6. Test Nginx for configuration errors.

CONSOLE

```
$ sudo nginx -t
```

Output:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok  
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

7. Restart Nginx to apply the Open WebUI virtual host configuration changes.

CONSOLE

```
$ sudo systemctl restart nginx
```

8. Allow HTTP connections through the firewall.

CONSOLE

```
$ sudo ufw allow http
```

Generate SSL Certificates to Secure Open WebUI

SSL certificates encrypt the connection between a client's web browser and your server. Use a trusted certificate authority (CA) like Let's Encrypt to generate SSL certificates and secure connections to the Open WebUI using your Nginx configuration. In the following steps, install the Certbot Let's Encrypt client and generate SSL certificates to access Open WebUI using your domain.

1. Install Certbot for Nginx.

CONSOLE

```
$ sudo apt install python3-certbot-nginx -y
```

2. Generate a new SSL certificate for your domain. Replace

`openwebui.example.com` with your domain and `admin@example.com` with an active email address.

CONSOLE

```
$ sudo certbot --nginx -d openwebui.example.com -m  
admin@example.com --agree-tos
```

Output:

```
Successfully received certificate.  
Certificate is saved at: /etc/letsencrypt/live/openwebui.example.com/  
fullchain.pem  
Key is saved at: /etc/letsencrypt/live/openwebui.example.com/privkey.pem  
This certificate expires on 2025-02-25.  
These files will be updated when the certificate renews.  
Certbot has set up a scheduled task to automatically renew this certificate in  
the background.  
  
Deploying certificate  
Successfully deployed certificate for openwebui.example.com to /etc/nginx/sites-  
enabled/openwebui.conf  
Congratulations! You have successfully enabled HTTPS on https://  
openwebui.example.com
```

- Restart Nginx to apply the SSL configuration changes.

CONSOLE

```
$ sudo systemctl restart nginx
```

- Allow HTTPS connections through the firewall.

CONSOLE

```
$ sudo ufw allow https
```

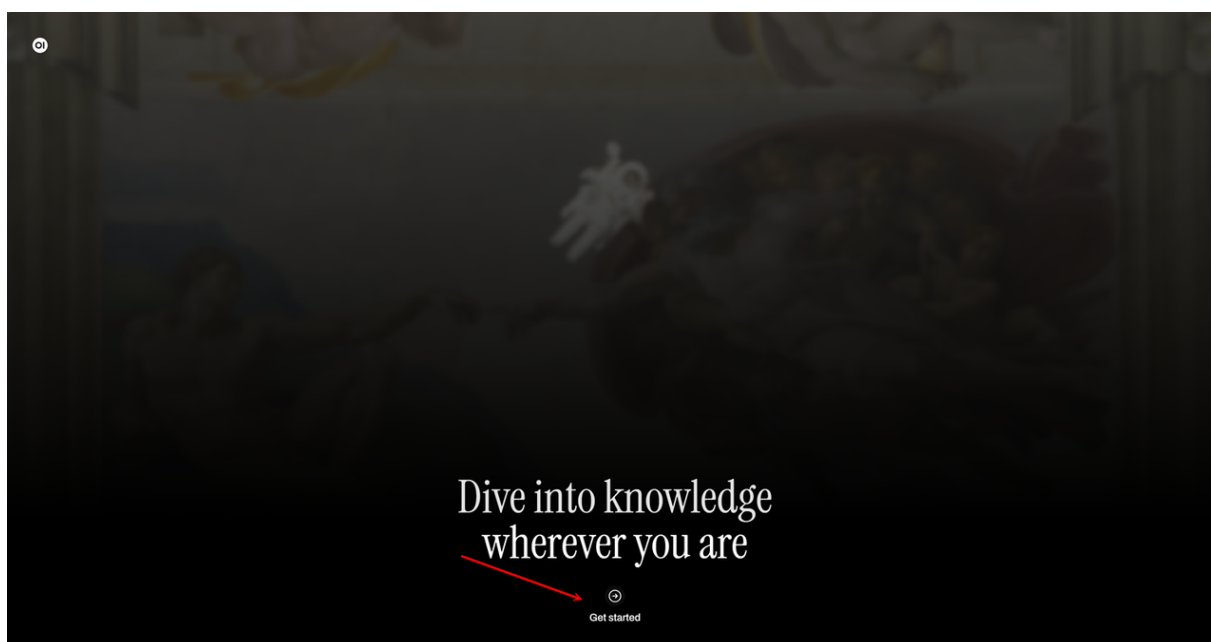
Access Open WebUI

You can access Open WebUI using your domain based on the configuration steps you performed earlier. In the following steps, access Open WebUI and create a new administrator account to run LLMs on the server.

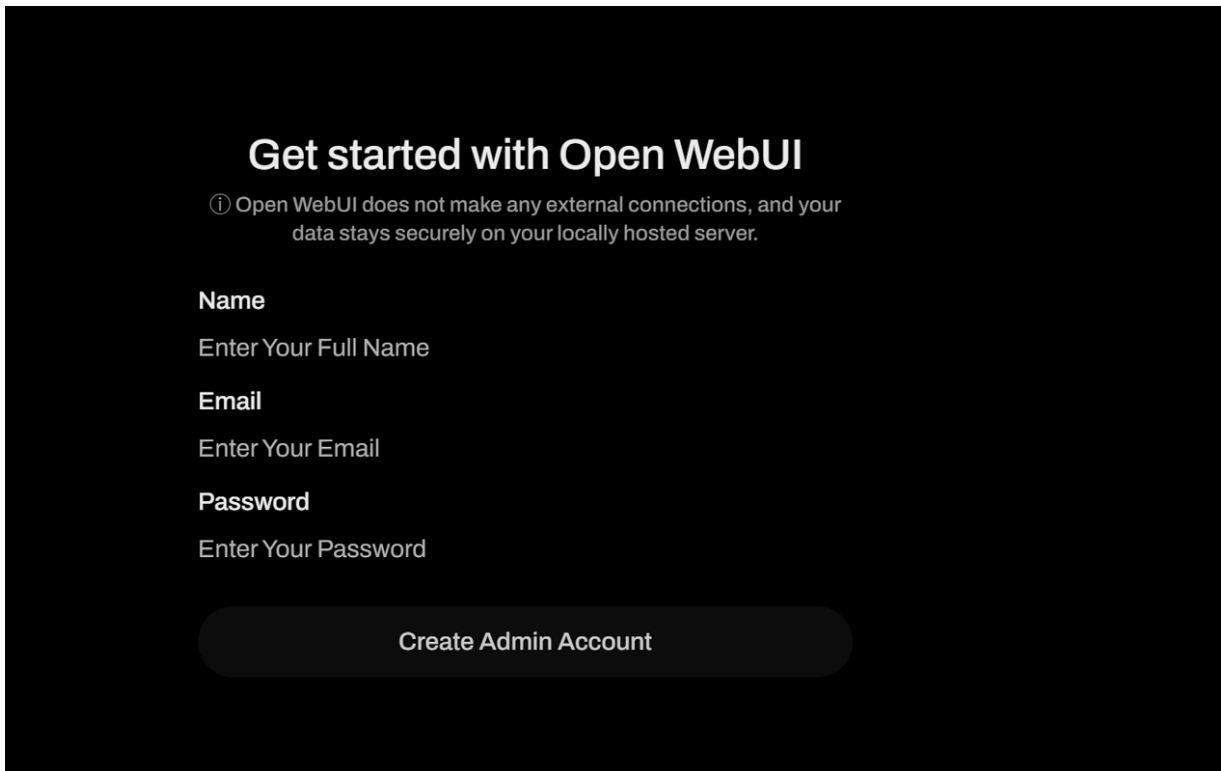
- Access your Open WebUI's domain to access the web interface.

```
https://openwebui.example.com
```

- Click **Get started** to access Open WebUI.



3. Enter a new administrator username, email address, and password in the Open WebUI fields.



Get started with Open WebUI

Open WebUI does not make any external connections, and your data stays securely on your locally hosted server.

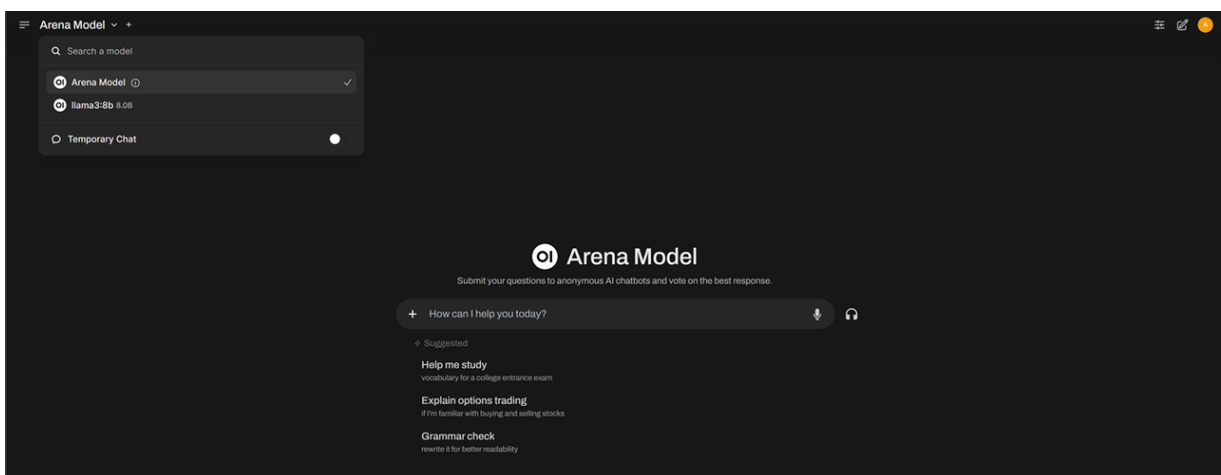
Name
Enter Your Full Name

Email
Enter Your Email

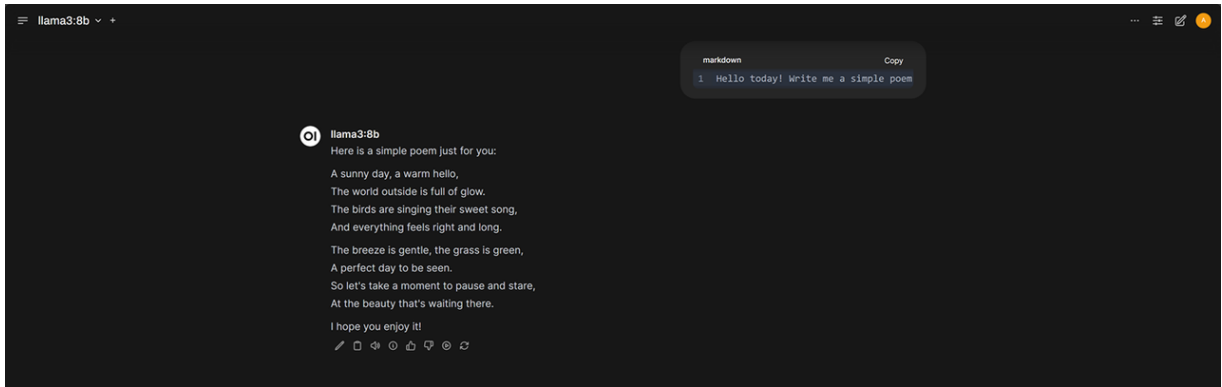
Password
Enter Your Password

Create Admin Account

4. Click **Create Admin Account** to create the first Open WebUI administrator account.
5. Confirm that the Open WebUI page displays in your browser, click the **Arena Model** drop-down in the top menu, and select the `Llama 8b` model you installed earlier.



6. Enter a new prompt like `Hello today! Write me a simple poem` to generate a result using the default `Llama 8b` model you installed earlier.



Install and Run LLMs in Open WebUI

Open WebUI ships with some default AI models. You can also install new models directly using Ollama on the server or by uploading the model files through the Open WebUI interface. In the following steps, install new models using Ollama and run them using Open WebUI.

1. [Visit the Ollama library](#) and find new models.
2. Download a new model. For example, the `mistral 7B` model.

CONSOLE

```
$ sudo ollama pull mistral
```

3. Download another model, such as `llama3:70b`.

CONSOLE

```
$ sudo ollama pull llama3:70b
```

The `llama3:70b` model is `40GB` and requires more RAM to run on the server.

4. Restart Open WebUI to synchronize the new model changes.

CONSOLE

```
$ sudo docker restart open-webui
```

Or, restart the Open WebUI system service.

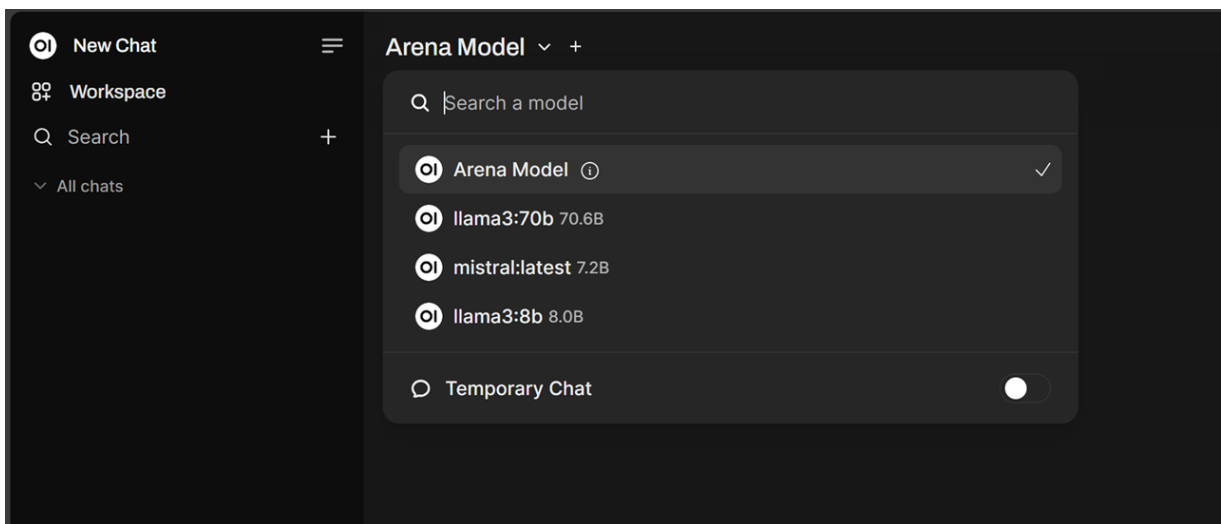
CONSOLE

```
$ sudo systemctl restart openwebui
```

5. Access Open WebUI in a new web browser.

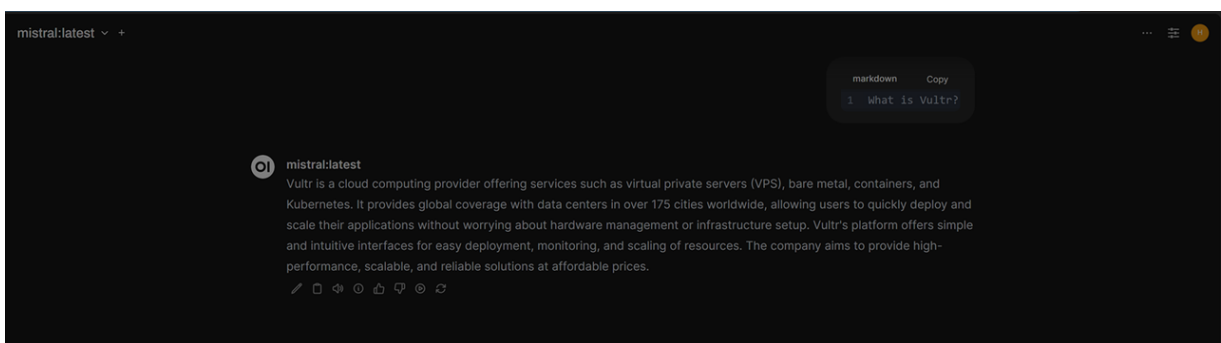
```
https://openwebui.example.com
```

6. Click the models list on the top left drop-down list and verify the new models are available.



7. Select a model to use in Open WebUI. For example, select the Mistral model you installed.

8. Enter a new prompt in the input field, such as `What is Vultr?` and verify that the model processes new results.



Conclusion

Open WebUI is a powerful web interface for running ML LLMs on a server. In this guide, you installed Open WebUI on a Vultr Cloud GPU instance and secured it with SSL certificates. You can customize the Open WebUI interface and create multiple users to log in or sign up to use LLMs directly on your server. For additional guidelines, please refer to the [Open WebUI documentation](#).



VULTR

