

# Setup YouTube Downloader Web Server -source code / Build Your Own

## Contents

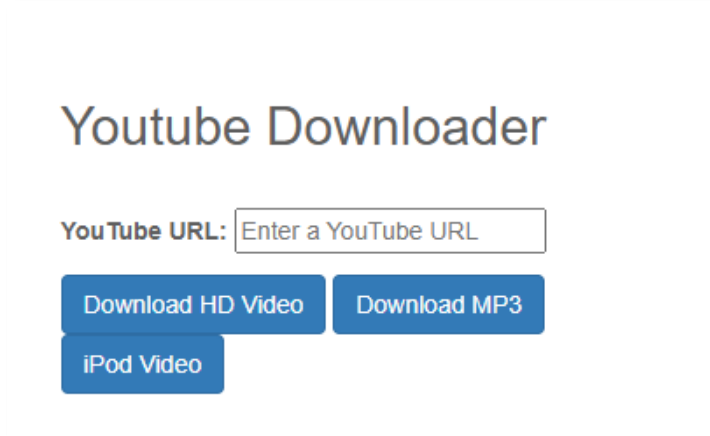
Introduction .....	2
Install the server .....	2
We are using Ubuntu server VM or Raspberry Pi OS and just need to enable SSH with full access ..	2
Install Django Framework and necessary Apps.....	3
Create virtual environment and install Django .....	3
Create web applications and install dependent libraries.....	4
Copy the source code files and install.....	4
Setup and configure the web server .....	6

## Introduction

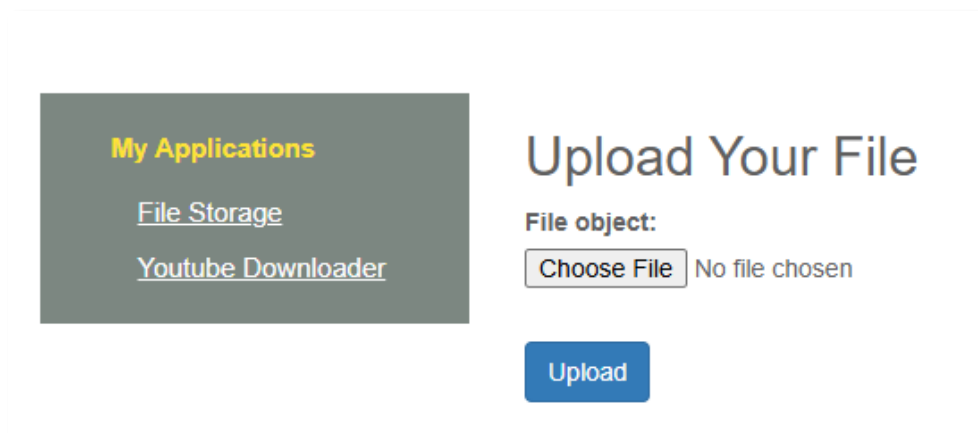
This is an open-source web app mainly for downloading your favourite YouTube video without using the public ones which contains a lot of annoying ads and fake downloads.

It can download 1080 HD video and 480p format for older generation iPods.

It can also just download the audio into MP3 format.



In addition, it has a simple online storage tool that allows you to upload / download your files from your phone, computer and share them in between devices. Your files will just be saved to your own server without paying 3<sup>rd</sup> party subscription and safe.



## Install the server

We are using Ubuntu server VM or Raspberry Pi OS and just need to enable SSH with full access

- Ubuntu server 24.04.01 (minimal footprint install / 2 CPU / 2G RAM / 20Gb Disk)
  - If the system running on VM. Make sure all disk space has been used.

```
sudo lvextend -l +100%FREE /dev/ubuntuvg/ubuntuvl
```

```
sudo resize2fs /dev/mapper/ubuntuvg-ubuntuvl
```

```
user@filesafer:~$ df -h
Filesystem                Size      Used Avail Use% Mounted on
tmpfs                      192M        976K  192M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 18G       4.0G    13G   24% /
tmpfs                      960M         0   960M   0% /dev/shm
tmpfs                      5.0M         0    5.0M   0% /run/lock
/dev/sda2                  1.8G       99M    1.6G   7% /boot
tmpfs                      192M       12K    192M   1% /run/user/1000
```

- Raspberry Pi 4b (2G RAM / 16Gb Micro SD)
  - Install Raspberry Pi OS Lite (no Desktop needed)

## Install Django Framework and necessary Apps

**sudo apt-get update**

**sudo apt-get install ffmpeg**

**sudo apt-get install python3-django python3-pip python3-venv**

- If you are using Ubuntu server 20.04 with minimal footprint installation

**sudo apt-get install nano cron iputils-ping**

**sudo nano /etc/resolv.conf**

add line: **nameserver 8.8.8.8**

## Create virtual environment and install Django

**cd /**

**sudo mkdir Automation**

**sudo chmod 777 Automation**

**python3 -m venv Automation**

**cd Automation**

**source bin/activate**

**pip3 install Django**

```

user@filesafer:~$ cd /
user@filesafer:/$ sudo mkdir Automation
user@filesafer:/$ sudo chmod 777 Automation/
user@filesafer:/$ python3 -m venv Automation
user@filesafer:/$ cd Automation
source bin/activate
pip3 install Django
Collecting Django
  Downloading django-5.2.2-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref>=3.8.1 (from Django)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from Django)
  Downloading sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
  Downloading django-5.2.2-py3-none-any.whl (8.3 MB)
    8.3/8.3 MB 6.3 MB/s eta 0:00:00
  Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
  Downloading sqlparse-0.5.3-py3-none-any.whl (44 kB)
    44.4/44.4 kB 3.8 MB/s eta 0:00:00
Installing collected packages: sqlparse, asgiref, Django
Successfully installed Django-5.2.2 asgiref-3.8.1 sqlparse-0.5.3
(Automation) user@filesafer:/Automation$
(Automation) user@filesafer:/Automation$

```

## Create web applications and install dependent libraries

- Perform below commands under directory /Automation within the virtual environment

**django-admin startproject Downloader .**

**python3 manage.py startapp Login**

**python3 manage.py startapp Efile**

**python3 manage.py startapp Youtube\_downloader**

**pip3 install ipcalc six yt\_dlp django-sslserver**

- Above process creates below folder structure

Downloader

Youtube\_downloader

Efile

Login

Youtube\_downloader

## Copy the source code files and install

We will use sftp to copy the source files into created directories from above

If you are using windows computer you can use MoxaXterm free application to do this. Linux user can perform this task natively using Terminal.

- Start from the directory where you have downloaded source files. SFTP to the server

**cd /Automation**

**put -R \***

- All files source files will be copied into corresponded folders. And Automation folder should looks like below

```
(Automation) user@filesafer:/Automation$ ls -l
total 56
drwxr-xr-x 2 user user 4096 Jun  9 02:56 Cert
drwxr-xr-x 3 user user 4096 Jun  9 02:43 Downloader
drwxr-xr-x 3 user user 4096 Jun  9 02:56 Efile
drwxr-xr-x 3 user user 4096 Jun  9 02:56 Login
drwxr-xr-x 4 user user 4096 Jun  9 02:56 Static
drwxr-xr-x 2 user user 4096 Jun  9 02:56 WebTemplates
drwxr-xr-x 3 user user 4096 Jun  9 02:56 Youtube_downloader
drwxrwxr-x 2 user user 4096 Jun  9 02:52 bin
drwxrwxr-x 3 user user 4096 Jun  9 02:34 include
drwxrwxr-x 3 user user 4096 Jun  9 02:34 lib
lrwxrwxrwx 1 user user    3 Jun  9 02:34 lib64 -> lib
-rwxrwxr-x 1 user user  666 Jun  9 02:42 manage.py
-rw-rw-r-- 1 user user  150 Jun  9 02:34 pyenv.cfg
-rwxr-xr-x 1 user user  542 Jun  9 02:56 run.sh
drwxrwxr-x 7 user user 4096 Jun  9 02:52 share
(Automation) user@filesafer:/Automation$
```

- Create Database

**python3 manage.py makemigrations**

**python3 manage.py migrate**

- Create superuser for the Web App. This account is for you to login to the Web GUI

**python3 manage.py createsuperuser**

```
(Automation) user@filesafer:/Automation$ python3 manage.py createsuperuser
Username (leave blank to use 'user'): user
Email address:
Password:
Password (again):
Superuser created successfully.
(Automation) user@filesafer:/Automation$
```

- Setup application auto-start (My ubuntu user account called "user" where used in @reboot code below)

**sudo nano /etc/crontab**

- add below line to the bottom and save / exit

**@reboot user /bin/bash -c "/Automation/run\_http.sh"**

- (Optional) run the Django Web app without front end web reverse proxy

Change above line to below and the web server will be ready

**https://<your-server-ip-address>:8000**

**@reboot user /bin/bash -c "/Automation/run.sh"**

## Setup and configure the web server

- Setup Nginx (Engine X) as front-end web server

- Install Nginx

```
sudo apt-get install nginx
```

- Setup web site in nginx by creating file “Django” in /etc/nginx/sites-available/

```
sudo nano /etc/nginx/sites-available/Django
```

- Copy below to the file, save and exit

```
server {  
  
listen 443 ssl;  
  
server_name <server FQDN or IP Address>;  
  
ssl_certificate /Automation/Cert/server.crt;  
  
ssl_certificate_key /Automation/Cert/server.key;  
  
  
location / {  
  
proxy_pass http://127.0.0.1:8000;  
  
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;  
  
proxy_read_timeout 600;  
  
proxy_connect_timeout 600;  
  
proxy_send_timeout 600;  
  
}  
  
}  
  
server {  
  
listen 80;  
  
server_name your.domain.or.ip;  
  
return 301 https://$host$request_uri;  
  
}
```

- Start Nginx

```
sudo ln -s /etc/nginx/sites-available/Django /etc/nginx/sites-enabled/
```

```
sudo nginx -t  
sudo systemctl restart nginx
```

- Reboot the server and access the web app via browser

**sudo reboot now**

**https://<your-server-ip-address>**

