# Video file: 80

# Code file : Setup Django Async Server ASGI using Uvicorn

**Below our own self doc**

**Setup Django Async Server (ASGI) using Uvicorn**

* **ASGI (Asynchronous Server Gateway Interface)** lets Django handle async tasks like WebSockets, long polling, and background tasks.
* **Uvicorn** is a high-performance ASGI server (based on **uvloop** and **httptools**) that runs Django asynchronously.
* It’s widely used in production for frameworks like **Django (ASGI mode)** and **FastAPI**.

**✅ Why Use Uvicorn?**

* Faster than Daphne for simple async HTTP workloads.
* Lightweight, easy to run, and integrates with Django ASGI.
* Supports WebSockets, HTTP/2, and async views.

**🔑 Setup Steps:**

1. **Install Uvicorn**

pip install uvicorn

1. **Enable ASGI in Django**  
   In settings.py:

ASGI\_APPLICATION = 'myproject.asgi.application'

1. **Check/Edit asgi.py** (already exists in Django ≥3.0 projects):

import os

from django.core.asgi import get\_asgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'myproject.settings')

application = get\_asgi\_application()

1. **Run server with Uvicorn**

uvicorn myproject.asgi:application --reload

Example with host & port:

uvicorn myproject.asgi:application --host 0.0.0.0 --port 8000

⚡ **Summary for Notes:**  
ASGI enables Django to handle async requests (like WebSockets). **Uvicorn** is a lightweight, high-speed ASGI server for running Django asynchronously. To set it up: install Uvicorn, configure ASGI\_APPLICATION, check asgi.py, and start Django with uvicorn myproject.asgi:application.

Self Documentation:

First run for installing the packege :

pip install uvicorn

after successfullly installing this run :

first time run :

uvicorn ch72.asgi:application

after that use this for running

uvicorn ch72.asgi:application –-reload

now our server can load

but you note that this command is very long so we can also make our own short command :

1. first make new app name core and in settings.py installed this app
2. in core application make one filder name as management/inside that make another folder named commands and inside that make \_\_init\_\_.py packege and also make the same folder inside the management
3. after than inside commands folder make another file name as runuvicorn.py

addthis code in runuvicorn.py:

from django.core.management.base import BaseCommand

import uvicorn

class Command(BaseCommand):

    help = "Run the Django Application with Uvicon"

    def handle(self,\*args,\*\*options):

        uvicorn.run("ch70.asgi:application",host="127.0.0.1",port=8000,reload=True)

1. now we have a new and short commands can be created :

python manage.py runuvicorn