📦 Install Required Packages

```
!pip install fastapi nest-asyncio uvicorn pyngrok scikit-learn
```

Show hidden output

```
YOUR_AUTHTOKEN = "2zqCNES4ml0fu0BSGn5vnJbVXJn_3x4WJjTGAqtHrhkyq1JdP"
```

Task 1

```
from fastapi import FastAPI
from pyngrok import ngrok
import uvicorn, nest_asyncio

app = FastAPI()

@app.get("/health")
def health():
    return {"status": "OK", "version": "1.0.0"}

nest_asyncio.apply()
ngrok.set_auth_token(YOUR_AUTHTOKEN)
public_url = ngrok.connect(8000)
print("Public URL:", public_url)
uvicorn.run(app, host="0.0.0.0", port=8000)
```

```
Public URL: NgrokTunnel: "https://5f7bf7c612f0.ngrok-free.app" -> "http://localhost:8
INFO:     Started server process [478]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [478]
```

1) FastAPI Health Check with ngrok - Line-by-Line Explanation

Code Breakdown

```
from fastapi import FastAPI
```

- **Imports `FastAPI`**: This is the core web framework used to build your API endpoints. It provides tools to define routes (like `/health`) and handle HTTP requests.

```
from pyngrok import ngrok
```

- **Imports `ngrok`**: This tool is essential for exposing your local server to the internet via a public URL. It's incredibly useful for temporary sharing or testing APIs without a full deployment.

```
import uvicorn, nest_asyncio
```

- **`uvicorn`**: A high-performance ASGI server responsible for running your FastAPI application.
- **`nest_asyncio`**: This library is crucial for environments like Google Colab or Jupyter Notebooks. These environments often have an active asyncio event loop already running. Without `nest_asyncio.apply()`, `uvicorn.run()` would likely throw a `RuntimeError` due to conflicting event loops.

```
app = FastAPI()
```

- **Initializes the FastAPI app**: This creates an instance of your FastAPI application. The `app` object is what you'll use to define all your API routes and configure your web server.

```
@app.get("/health")
def health():
    return {"status": "OK", "version": "1.0.0"}
```

- **Defines a GET endpoint at `/health`**:
  - `@app.get("/health")` is a decorator that registers the `health` function to handle HTTP GET requests to the `/health` path.
  - When a client (e.g., a web browser, `curl`, Postman) sends a GET request to `/health`, this function executes.

  - It returns a JSON response:

    JSON

    ```
    {
      "status": "OK",
      "version": "1.0.0"
    }
    ```

  - This is a standard practice for a **"health check" endpoint**, used to confirm that your server is operational and responding as expected.

```
nest_asyncio.apply()
```

- **Applies `nest_asyncio` patch**: This line applies the necessary patch to allow `uvicorn.run()` to operate correctly within interactive environments like Colab or Jupyter, preventing conflicts with existing asynchronous event loops.

```
ngrok.set_auth_token(YOUR_AUTHTOKEN)
```

- **Sets ngrok authentication token**:
  - **Important**: Replace `YOUR_AUTHTOKEN` with your actual ngrok authentication token.
  - You can obtain your token from your ngrok dashboard: https://dashboard.ngrok.com/get-started/your-authtoken.
  - This token is required to programmatically use ngrok services and establish tunnels.

```
public_url = ngrok.connect(8000)
```

- **Opens a public ngrok tunnel**:
  - This command establishes a secure public tunnel from the ngrok cloud service to your local server, which will be running on **port `8000`**.
  - The `public_url` variable will store the generated public URL (e.g., `https://abc123.ngrok-free.app`).
  - You can share this URL with anyone to allow them to access your FastAPI application from anywhere on the internet.

```
print("Public URL:", public_url)
```

- **Displays the public URL**: This line simply prints the generated `public_url` to your Colab output, making it easy for you to see and access your live API.

```
uvicorn.run(app, host="0.0.0.0", port=8000)
```

- **Starts the FastAPI server with uvicorn**:
  - `uvicorn.run(app, ...)` starts the Uvicorn server, running your `app` (the FastAPI instance).
  - `host="0.0.0.0"`: This setting makes your server accessible from any IP address. If you set it to `127.0.0.1` or `localhost`, it would only be accessible from the machine where it's running. For ngrok to connect, it needs to be `0.0.0.0`.
  - `port=8000`: This specifies the local port on which your FastAPI server will listen for incoming requests. This port must match the port specified in `ngrok.connect()`.

🌐 Result: Publicly Accessible API

After running the code, you will get a publicly accessible API at a URL similar to:

```
https://abc123.ngrok-free.app/health
```

When you open this URL in your browser or use a tool like `curl`, you will see the JSON response:

```
{
  "status": "OK",
  "version": "1.0.0"
}
```

TASK 2

```
#The relationship here is perfectly linear:
#y = 2 * x
#curl -X POST "<your-ngrok-url>/predict" -H "Content-Type: application/json" -d '{"x": 5}'

#🧠 Train Model → 🚀 Expose with FastAPI → 🌎 Publish with ngrok → 📥 POST JSON → 🎯 Ret

from fastapi import FastAPI
from pydantic import BaseModel
from sklearn.linear_model import LinearRegression
import numpy as np
from pyngrok import ngrok
import uvicorn
import nest_asyncio

# Training a simple linear regression model
X = np.array([[1], [2], [3]])
y = np.array([2, 4, 6])
model = LinearRegression().fit(X, y)

app = FastAPI()

class Input(BaseModel):
    x: float

@app.post("/predict")
def predict(data: Input):
    prediction = model.predict(np.array([[data.x]]))[0]
    return {"prediction": prediction}

# Needed to allow nested event loops (for Jupyter and scripts)
nest_asyncio.apply()

ngrok.set_auth_token(YOUR_AUTHTOKEN)

# Open tunnel
public_url = ngrok.connect(8000)
print("Public URL:", public_url)

# Run FastAPI app
uvicorn.run(app, host="0.0.0.0", port=8000)
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
/tmp/ipython-input-1-450596186.py in <cell line: 0>()
      9 from sklearn.linear_model import LinearRegression
     10 import numpy as np
---> 11 from pyngrok import ngrok
     12 import uvicorn
     13 import nest_asyncio

ModuleNotFoundError: No module named 'pyngrok'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
```

OPEN EXAMPLES

🧠 Linear Regression FastAPI App — Line-by-Line

python

`# The relationship here is perfectly linear: # y = 2 * x`

- Comment explaining that the training data follows a simple linear relationship (double of input).

`# curl -X POST "<your-ngrok-url>/predict" -H "Content-Type: application/json" -d '{"x": 5}'`

- Example `curl` command to test the `/predict` endpoint with input `x = 5`.

`# 🧠 Train Model → 🚀 Expose with FastAPI → 🌍 Publish with ngrok → 📥 POST JSON → 🎯 Return Prediction`

- Summary of the pipeline: Train → Expose API → Publish → Use

---

📦 Imports

`from fastapi import FastAPI`

- Imports `FastAPI`, the core web framework used to build APIs.

`from pydantic import BaseModel`

- Imports `BaseModel` from Pydantic for request validation and type enforcement.

`from sklearn.linear_model import LinearRegression`

- Imports `LinearRegression` from scikit-learn for training a basic ML model.

`import numpy as np`

- Imports NumPy for numerical operations and array handling.

`from pyngrok import ngrok`

- Imports `ngrok` to expose the local server to the internet via a public tunnel.

`import uvicorn import nest_asyncio`

- `uvicorn` runs the FastAPI app as an ASGI server.
- `nest_asyncio` allows `uvicorn.run()` to execute in environments like Google Colab or

Jupyter, where an event loop is already running.

🔧 Train the ML Model

`X = np.array([[1], [2], [3]]) y = np.array([2, 4, 6]) model = LinearRegression().fit(X, y)`

- `X` is the input data and `y` is the target output.
- This trains a model where the learned function is essentially:

```
`y = 2 * x`
```

⚙️ Initialize FastAPI App

`app = FastAPI()`

- Creates an instance of a FastAPI app where you can define API routes.

📥 Define Input Model

`class Input(BaseModel): x: float`

- Defines the shape of incoming request data as a Pydantic model.
- The API expects a JSON payload like: `{"x": 5}`

🔮 Define Prediction Endpoint

`@app.post("/predict") def predict(data: Input): prediction = model.predict(np.array([[data.x]]))[0] return {"prediction": prediction}`

- `@app.post("/predict")`: Declares a POST endpoint `/predict`.
- `data: Input`: Automatically parses and validates incoming JSON using `Input` model.
- `model.predict(...)`: Makes a prediction using the trained model.
- `return`: Sends back the result as JSON like `{"prediction": 10.0}`

🔁 Asyncio Patch for Colab

`nest_asyncio.apply()`

- Applies a patch to let `uvicorn.run()` work inside notebooks that already use an asyncio event loop.

🔐 Set ngrok Auth Token

`ngrok.set_auth_token(YOUR_AUTHTOKEN)`

- Authorizes ngrok to create public tunnels.
- Replace `YOUR_AUTHTOKEN` with your token from: https://dashboard.ngrok.com

---

🌍 Open Public Tunnel

`public_url = ngrok.connect(8000) print("Public URL:", public_url)`

- Exposes your local FastAPI server (port 8000) to the internet.
- Prints the public URL so you can send requests like `curl https://abc123.ngrok-free.app/predict`

🚀 Run the FastAPI App

`uvicorn.run(app, host="0.0.0.0", port=8000)`

- Starts the FastAPI app using `uvicorn`.
- `host="0.0.0.0"` allows access from any device (needed for ngrok).
- `port=8000` is the port ngrok connects to.

✅ Example Usage

```
curl -X POST "https://your-ngrok-url.ngrok-free.app/predict"
-H "Content-Type: application/json"
-d '{"x": 5}'
```

Response:

```
{ "prediction": 10.0 }
```

3) Text Cleaning service - Fast API

```
"""
curl -X POST "https://dfdf3e66099c.ngrok-free.app/clean"
-H "Content-Type: application/json"
-d '{"text": "Hello, World! This is @FastAPI...#2025"}'
"""

from fastapi import FastAPI
from pydantic import BaseModel
import re
from pyngrok import ngrok
import uvicorn, nest_asyncio

app = FastAPI()

class Input(BaseModel):
    text: str

@app.post("/clean")
def clean_text(data: Input):
    cleaned = re.sub(r"[^\w\s]", "", data.text.lower())
    return {"cleaned_text": cleaned}

nest_asyncio.apply()
ngrok.set_auth_token(YOUR_AUTHTOKEN)
public_url = ngrok.connect(8000)
print("Public URL:", public_url)
uvicorn.run(app, host="0.0.0.0", port=8000)
```

```
Public URL: NgrokTunnel: "https://479d257021c2.ngrok-free.app" -> "http://localhost:8
INFO:     Started server process [478]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [478]
```

📈 4. Logging for Monitoring

```
from fastapi import FastAPI, Request
from pyngrok import ngrok
import uvicorn, nest_asyncio
import datetime

app = FastAPI()

@app.get("/ping")
async def ping(request: Request):
    log = f"{datetime.datetime.now()} - Ping from {request.client.host}"
    print(log)
    return {"message": "pong"}

nest_asyncio.apply()
```

```
ngrok.set_auth_token(YOUR_AUTHTOKEN)
public_url = ngrok.connect(8000)
print("Public URL:", public_url)
uvicorn.run(app, host="0.0.0.0", port=8000)
```

```
Public URL: NgrokTunnel: "https://c15c0e24334b.ngrok-free.app" -> "http://localhost:8
INFO:     Started server process [478]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
ERROR:asyncio:Task exception was never retrieved
future: <Task finished name='Task-15' coro=<Server.serve() done, defined at /usr/loca
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/uvicorn/main.py", line 580, in run
    server.run()
  File "/usr/local/lib/python3.11/dist-packages/uvicorn/server.py", line 67, in run
    return asyncio.run(self.serve(sockets=sockets))
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/nest_asyncio.py", line 30, in run
    return loop.run_until_complete(task)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/nest_asyncio.py", line 92, in run_unt
    self._run_once()
  File "/usr/local/lib/python3.11/dist-packages/nest_asyncio.py", line 133, in _run_o
    handle._run()
  File "/usr/lib/python3.11/asyncio/events.py", line 84, in _run
    self._context.run(self._callback, *self._args)
  File "/usr/lib/python3.11/asyncio/tasks.py", line 360, in __wakeup
    self.__step()
  File "/usr/lib/python3.11/asyncio/tasks.py", line 277, in __step
    result = coro.send(None)
             ^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/uvicorn/server.py", line 70, in serve
    with self.capture_signals():
  File "/usr/lib/python3.11/contextlib.py", line 144, in __exit__
    next(self.gen)
  File "/usr/local/lib/python3.11/dist-packages/uvicorn/server.py", line 331, in capt
    signal.raise_signal(captured_signal)
KeyboardInterrupt
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET / HTTP/1.1" 404 Not Found
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /favicon.ico HTTP/1.1" 404
2025-07-13 23:51:06.479626 - Ping from 2401:4900:67b9:4a94:c442:a314:d0c5:2261
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /ping HTTP/1.1" 200 OK
2025-07-13 23:51:07.056793 - Ping from 2401:4900:67b9:4a94:c442:a314:d0c5:2261
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /ping HTTP/1.1" 200 OK
2025-07-13 23:51:07.379357 - Ping from 2401:4900:67b9:4a94:c442:a314:d0c5:2261
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /ping HTTP/1.1" 200 OK
WARNING:pyngrok.process.ngrok:t=2025-07-13T23:51:23+0000 lvl=warn msg="Stopping forwa
WARNING:pyngrok.process.ngrok:t=2025-07-13T23:51:23+0000 lvl=warn msg="Error restarti
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [478]
```

🤖 5. Readiness + Fake Prediction

```
from fastapi import FastAPI
from pyngrok import ngrok
import uvicorn, nest_asyncio
import random

app = FastAPI()

@app.get("/ready")
def readiness():
    return {"ready": True}

@app.get("/predict")
def fake_prediction():
    return {"result": random.choice(["cat", "dog", "rabbit"])}

nest_asyncio.apply()
ngrok.set_auth_token(YOUR_AUTHTOKEN)
public_url = ngrok.connect(8000)
print("Public URL:", public_url)
uvicorn.run(app, host="0.0.0.0", port=8000)
```

```
Public URL: NgrokTunnel: "https://7999869f9aa0.ngrok-free.app" -> "http://localhost:8
INFO:     Started server process [478]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET / HTTP/1.1" 404 Not Found
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /favicon.ico HTTP/1.1" 404
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /predict HTTP/1.1" 200 OK
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /predict HTTP/1.1" 200 OK
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /predict HTTP/1.1" 200 OK
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /predict HTTP/1.1" 200 OK
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /predict HTTP/1.1" 200 OK
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /predict HTTP/1.1" 200 OK
INFO:     2401:4900:67b9:4a94:c442:a314:d0c5:2261:0 - "GET /predict HTTP/1.1" 200 OK
INFO:     Shutting down
INFO:     Waiting for application shutdown.
INFO:     Application shutdown complete.
INFO:     Finished server process [478]
```