# GIỚI THIỆU

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ, nhu cầu sở hữu một website riêng đã trở thành yêu cầu thiết yếu đối với cá nhân và doanh nghiệp. Website không chỉ đóng vai trò là kênh giới thiệu sản phẩm, dịch vụ và hình ảnh doanh nghiệp, mà còn là nền tảng để triển khai các ứng dụng nội bộ dưới dạng web application, góp phần nâng cao hiệu quả vận hành và năng lực cạnh tranh.

Để xây dựng một hệ thống website hoàn chỉnh, việc lựa chọn nền tảng hạ tầng phù hợp là yếu tố then chốt. Hiện nay, các hệ điều hành phổ biến bao gồm **Windows Server** và **Linux**, trong đó Linux cung cấp nhiều bản phân phối như Ubuntu, CentOS, Red Hat hay Fedora. Trong số đó, **Ubuntu** nổi bật nhờ tính ổn định, hiệu năng cao, khả năng đáp ứng đa dạng nhu cầu triển khai và đặc biệt là hoàn toàn miễn phí.

Bên cạnh hệ điều hành, lựa chọn phần cứng và mô hình triển khai cũng đóng vai trò quan trọng. Người dùng có thể đầu tư máy chủ vật lý hoặc sử dụng dịch vụ thuê máy chủ từ các nhà cung cấp đám mây. Trong số các giải pháp hiện có, **AWS Lightsail** được đánh giá là một lựa chọn nổi bật nhờ sự ổn định, hiệu suất tốt, chi phí hợp lý và cố định, ít thay đổi.

Về công nghệ phát triển website, có nhiều hướng tiếp cận khác nhau tùy theo mục tiêu và quy mô dự án. Tuy nhiên, sự kết hợp giữa **Nginx** và **Django** được xem là một giải pháp hiệu quả, linh hoạt và tối ưu về hiệu năng, phù hợp để xây dựng các ứng dụng web từ cơ bản đến thương mại.

Cuốn ebook này được biên soạn nhằm cung cấp hướng dẫn chi tiết và thực tiễn về cách thiết lập và triển khai hệ thống server dựa trên các lựa chọn nêu trên. Nội dung hướng tới việc giúp người đọc nắm vững quy trình từ cơ bản đến nâng cao, từ đó có thể tự tin xây dựng và vận hành một hệ thống website hoàn chỉnh, ổn định và hiệu quả.

# Chương 1: Làm quen với Linux

Chương này sẽ giúp bạn làm quen với những khái niệm cơ bản nhất của Linux – hệ điều hành thường được sử dụng để vận hành các máy chủ web. Thông qua nội dung đơn giản và dễ hiểu, bạn sẽ nắm được cách Linux hoạt động, cách tương tác với hệ thống và những thành phần quan trọng cần biết trước khi bắt đầu cài đặt và sử dụng server trong các chương tiếp theo.

## Nội dung

1. Hệ điều hành là gì ?
2. Giao diện Console và GUI khác nhau như thế nào ?
3. Sự khác biệt giữa Windows và Linux
4. Vì sao Linux phù hợp để làm server ?
5. Distro là gì? Vì sao Ubuntu, CentOS, Red Hat được gọi là distro?
6. Vì sao Ubuntu là lựa chọn phổ biến hiện nay?
7. SSH là gì?
8. SSH key pair là gì? Public key và private key dùng để làm gì?
9. Port và Firewall là gì?
10. Cron job và crontab là gì?
11. Systemd, systemctl, apt và apt-get là gì?
12. User, Group, Other và quyền của mỗi file trên Linux.
13. Chmod và chown là gì?

# Hệ điều hành là gì ?

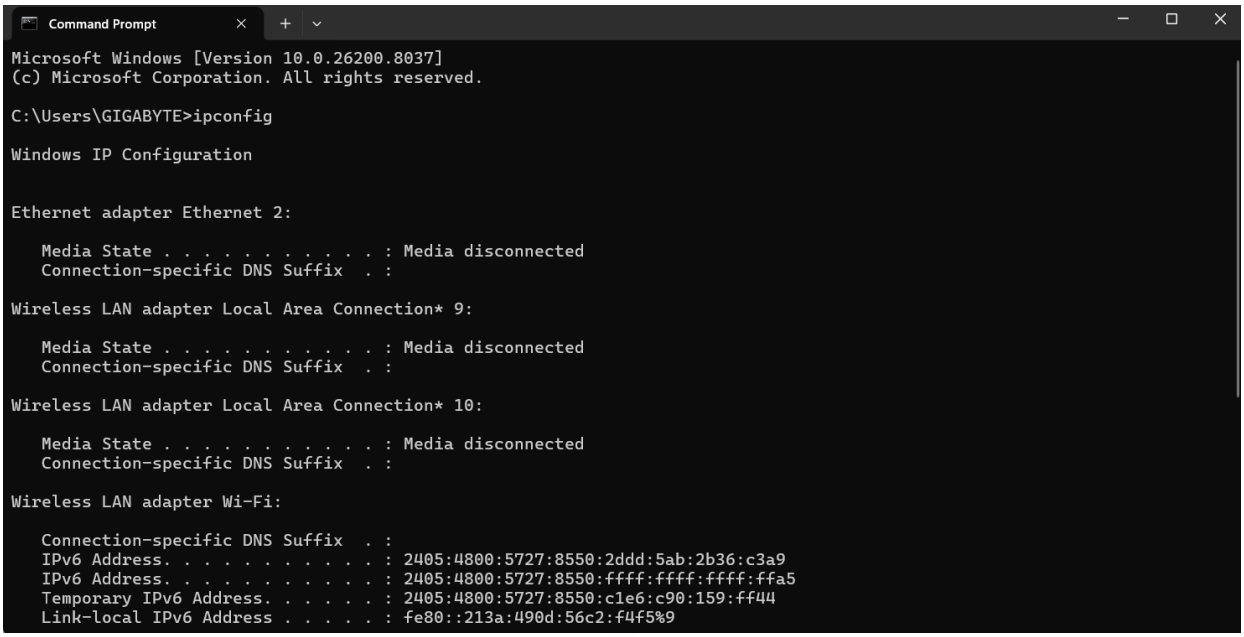
**Hệ điều hành** là một loại phần mềm hệ thống đóng vai trò trung gian giữa người dùng, các ứng dụng và phần cứng của máy tính. Khi bạn thực hiện các thao tác như mở ứng dụng, lưu file, truy cập dữ liệu, hay kết nối mạng, hệ điều hành sẽ tiếp nhận các yêu cầu này, sau đó quản lý và điều phối tài nguyên phần cứng (như CPU, bộ nhớ, ổ đĩa, thiết bị ngoại vi) để đảm bảo các chương trình hoạt động đúng cách, hiệu quả và không xung đột với nhau.

# Giao diện Console và GUI khác nhau như thế nào ?

Khi sử dụng máy tính, bạn có thể tương tác theo hai cách phổ biến: dùng giao diện đồ họa (GUI) hoặc dùng dòng lệnh (Console/CLI). Hai cách này khác nhau chủ yếu ở cách bạn thao tác với máy tính.

**Console** là cách bạn điều khiển máy tính bằng cách gõ lệnh bằng chữ.

Ví dụ: Gõ lệnh ipconfig để kiểm tra cấu hình ip




**GUI** là cách bạn tương tác với máy tính thông qua hình ảnh trực quan như: Cửa sổ, nút bấm, biểu tượng , menu, ...

Ví dụ: bạn mở file bằng cách click chuột thay vì gõ lệnh.

## Một số sự khác biệt giữa Windows và Linux

Windows và Linux có nhiều điểm khác nhau trong cách tổ chức và vận hành hệ thống. Dưới đây là một số khác biệt cơ bản giúp bạn dễ hình dung:

- **Đường dẫn hệ thống**: Windows sử dụng ký tự \ (backslash) để phân tách các thư mục trong đường dẫn, ví dụ "*C:\Users\Documents*". Trong khi đó, Linux sử dụng ký tự / (forward slash) để phân tách thư mục trong một cây thư mục duy nhất bắt đầu từ /, ví dụ "*/home/user*". Hai ký tự này có cùng mục đích là giúp xác định cấu trúc đường dẫn giữa các thư mục, nhưng được quy ước khác nhau tùy theo hệ điều hành.
- **Cài đặt phần mềm**: Windows thường cài bằng file .exe thông qua giao diện; Linux cài bằng trình quản lý gói như apt, yum,... thông qua dòng lệnh.
- **Giao diện sử dụng**: Cả Windows và Linux đều hỗ trợ cả GUI (giao diện đồ họa) và Console (dòng lệnh). Tuy nhiên, trong thực tế Windows thường được sử dụng chủ yếu thông qua GUI với các thao tác trực quan bằng chuột và cửa sổ, trong khi Linux thường được sử dụng nhiều qua Console, đặc biệt là trên các hệ thống server, nơi việc thao tác bằng dòng lệnh giúp quản lý hệ thống nhanh chóng và hiệu quả hơn.
- **Tài nguyên sử dụng**: Windows thường tiêu tốn nhiều tài nguyên hơn; Linux nhẹ hơn, phù hợp chạy trên máy cấu hình thấp hoặc nhiều dịch vụ cùng lúc.
- **Mục đích sử dụng phổ biến**: Windows thường dùng cho máy cá nhân, làm việc văn phòng, giải trí; Linux phổ biến trong môi trường server, cloud và hệ thống backend.

## Vì sao Linux phù hợp làm Server

Linux được sử dụng rất phổ biến trong môi trường server nhờ vào nhiều ưu điểm phù hợp với yêu cầu vận hành hệ thống lâu dài, ổn định và hiệu quả. Dưới đây là những lý do chính:

- **Ổn định cao**: Linux có khả năng hoạt động liên tục trong thời gian dài mà ít gặp lỗi hoặc cần khởi động lại. Điều này rất quan trọng với server vì hệ thống thường phải chạy 24/7.
- **Tiết kiệm tài nguyên**: Linux nhẹ hơn so với nhiều hệ điều hành khác, không tiêu tốn quá nhiều RAM hay CPU. Điều này giúp server có thể xử lý nhiều yêu cầu cùng lúc mà không bị quá tải nhanh.
- **Bảo mật tốt**: Linux có cơ chế phân quyền chặt chẽ và được cộng đồng phát hiện, vá lỗi nhanh chóng. Điều này giúp giảm nguy cơ bị tấn công và tăng độ an toàn cho hệ thống.
- **Quản lý từ xa dễ dàng**: Linux hỗ trợ tốt các công cụ như SSH, cho phép quản trị server từ xa thông qua dòng lệnh một cách nhanh chóng và hiệu quả.
- **Hỗ trợ mạnh cho tự động hóa**: Các công cụ như cron, script shell, và các trình quản lý dịch vụ giúp dễ dàng tự động hóa các tác vụ như backup, cập nhật, hoặc chạy job định kỳ.
- **Mã nguồn mở và linh hoạt**: Linux cho phép tùy chỉnh sâu theo nhu cầu sử dụng, phù hợp với nhiều loại server khác nhau như web server, database server, hay application server.

## Distro là gì? Vì sao Ubuntu, CentOS, Red Hat,... được gọi là Distro?

**Distro:** một phiên bản Linux được đóng gói và phân phối sẵn, bao gồm nhân Linux (kernel) cùng với các phần mềm, công cụ hệ thống,... và giao diện được cấu hình sẵn để người dùng có thể sử dụng ngay.

Các hệ điều hành như Ubuntu, CentOS hay Red Hat không phải là "Linux nguyên bản", mà là những bản phân phối được xây dựng dựa trên Linux. Mỗi distro sẽ có:

- Cách cài đặt riêng
- Công cụ quản lý phần mềm riêng
- Cấu hình mặc định riêng
- Mục tiêu sử dụng khác nhau

Ví dụ:

- Ubuntu thường được tối ưu cho sự dễ dùng và phổ biến
- Red Hat hướng đến doanh nghiệp với hỗ trợ chính thức
- CentOS thường được dùng như bản miễn phí tương thích với Red Hat

## Vì sao Ubuntu là lựa chọn phổ biến hiện nay ?

Ubuntu là một trong những bản phân phối Linux được sử dụng rộng rãi, đặc biệt trong môi trường server và phát triển phần mềm, nhờ vào những ưu điểm sau:

- **Dễ cài đặt và sử dụng**: Ubuntu có quy trình cài đặt đơn giản, giao diện thân thiện, giúp người mới có thể làm quen nhanh chóng mà không cần nhiều kiến thức kỹ thuật ban đầu.
- **Cộng đồng lớn và tài liệu phong phú**: Ubuntu có lượng người dùng đông đảo trên toàn thế giới, vì vậy khi gặp vấn đề, bạn có thể dễ dàng tìm thấy hướng dẫn, tài liệu và giải pháp từ cộng đồng.
- **Hỗ trợ tốt từ Canonical**: Ubuntu được phát triển và duy trì bởi Canonical, với các phiên bản LTS (Long Term Support) được hỗ trợ lâu dài, rất phù hợp cho server cần sự ổn định.
- **Kho phần mềm phong phú**: Ubuntu có hệ thống quản lý gói apt và kho phần mềm lớn, giúp việc cài đặt, cập nhật và quản lý ứng dụng trở nên đơn giản và nhanh chóng.

- **Tương thích cao**: Ubuntu được hỗ trợ rộng rãi bởi nhiều phần mềm, framework và dịch vụ phổ biến, nên dễ triển khai trong các hệ thống thực tế.
- **Phổ biến trong môi trường cloud và server**: Nhiều nền tảng cloud (như AWS, Azure, GCP) đều hỗ trợ Ubuntu mặc định, giúp việc triển khai và vận hành hệ thống thuận tiện hơn.

## SSH là gì ?

**SSH** (Secure Shell) là một phương thức cho phép bạn truy cập và điều khiển máy tính từ xa một cách an toàn thông qua mạng internet.

Thay vì phải ngồi trực tiếp trước máy server, bạn có thể dùng SSH để kết nối từ máy cá nhân đến server và thực hiện các thao tác như:

- Chạy lệnh
- Cài đặt phần mềm
- Quản lý file và dịch vụ
- Kiểm tra hệ thống

SSH hoạt động chủ yếu qua giao diện dòng lệnh (Console), giúp quản trị hệ thống từ xa một cách nhanh chóng và hiệu quả.

Một điểm quan trọng của SSH là dữ liệu được mã hóa trong quá trình truyền, nên thông tin trao đổi giữa máy của bạn và server được bảo mật, tránh bị nghe lén hoặc đánh cắp.

## SSH key pair là gì? Public key và Private key dùng để làm gì?

**SSH key pair** là một cặp khóa dùng để xác thực khi kết nối SSH, giúp đăng nhập vào server một cách an toàn mà không cần nhập mật khẩu. Cơ chế này dựa trên mã hóa bất đối xứng (asymmetric cryptography), nghĩa là sử dụng hai khóa có liên quan với nhau nhưng khác nhau về chức năng: một khóa dùng để mã hóa và một khóa dùng để giải mã.

Một SSH key pair gồm 2 phần:

- **Private Key:** Đây là khóa được lưu trên máy cá nhân của bạn. Bạn phải giữ bí mật tuyệt đối và không chia sẻ cho người khác. Private key được dùng để "chứng minh danh tính" khi kết nối.
- **Public key:** Đây là khóa được đặt trên server. Bạn có thể chia sẻ public key mà không lo lộ thông tin bảo mật, vì nó chỉ dùng để kiểm tra và không thể suy ra private key từ public key.

Khi bạn kết nối SSH đến server :

1. Server sẽ kiểm tra public key đã lưu
2. Máy của bạn sử dụng private key để chứng minh danh tính
3. Server dùng public key tương ứng để kiểm tra chữ ký đó.
4. Nếu hai khóa khớp nhau thì bạn được phép đăng nhập

## Port và Firewall là gì ?

**Port** là một "điểm kết nối" trên máy tính dùng để phân biệt các dịch vụ đang chạy. Mỗi dịch vụ trên server sẽ hoạt động trên một port riêng. Có thể hiểu đơn giản: IP là địa chỉ của máy, còn port là "cửa" để đi vào từng dịch vụ cụ thể bên trong máy đó.

Ví dụ: Web server thường dùng port 80 (HTTP) hoặc 443 (HTTPS)

**Firewall** là một cơ chế giúp kiểm soát và bảo vệ kết nối mạng bằng cách cho phép hoặc chặn các truy cập vào/ra hệ thống. Firewall sẽ:

- Cho phép hoặc chặn truy cập vào các port cụ thể
- Ngăn các kết nối không mong muốn hoặc nguy hiểm
- Tăng mức độ bảo mật cho server

Ví dụ: Cho phép truy cập port 22 để SSH

## Cron job và Crontab là gì

**Cron job** là các tác vụ được hệ thống tự động chạy theo lịch định sẵn, thay vì bạn phải thực hiện thủ công.

Ví dụ :

- Tự động backup dữ liệu mỗi ngày lúc 2 giờ sáng
- Tự động gửi báo cáo vào mỗi tuần
- Tự động dọn file tạm sau một khoảng thời gian

**Crontab** là nơi để bạn khai báo và quản lý các cron job. Đây là một file hoặc công cụ dùng để định nghĩa: Khi nào cron job chạy và lệnh nào sẽ được thực thi. Bạn có thể hiểu crontab như một "bảng lịch" chứa danh sách các cron job.

## Systemd, systemctl, apt và apt-get là gì ?

**Systemd** là một hệ thống dùng để quản lý và điều khiển các dịch vụ (services) trên Linux. Nó chịu trách nhiệm khởi động hệ thống và đảm bảo các dịch vụ như web server, database... luôn chạy ổn định.

**Systemctl** là công cụ dòng lệnh dùng để tương tác với systemd. Bạn có thể dùng systemctl để:

- Khởi động một dịch vụ
- Dừng dịch vụ
- Kiểm tra trạng thái dịch vụ
- Bật/tắt dịch vụ khi hệ thống khởi động

Ví dụ: dùng systemctl để quản lý web server hoặc SSH service.

**Apt** là trình quản lý gói phần mềm trên Linux (thường dùng trên Ubuntu), giúp bạn:

- Cài đặt phần mềm
- Cập nhật phần mềm
- Gỡ bỏ phần mềm

**Apt-get** cũng là một công cụ quản lý gói tương tự apt, được sử dụng từ trước. Hiện nay apt là phiên bản hiện đại, thân thiện hơn với người dùng nhưng apt-get vẫn tồn tại để đảm bảo tương thích và dùng trong một số trường hợp kỹ thuật.

## User, Group, Other và quyền của mỗi file trên Linux

**User** là một tài khoản người dùng trên hệ thống. Mỗi người dùng có thể đăng nhập và thực hiện các thao tác tùy theo quyền được cấp.

Ví dụ: một server có thể có nhiều user khác nhau như user quản trị và user thông thường.

**Group** là một nhóm gồm nhiều user. Việc dùng group giúp quản lý quyền truy cập dễ dàng hơn khi nhiều người cần cùng truy cập vào một tài nguyên.

Ví dụ: Một nhóm "developer" có thể được cấp quyền truy cập vào cùng một thư mục dự án.

**Others** là những người dùng còn lại trên hệ thống, không thuộc về user sở hữu file (owner) và cũng không nằm trong group được gán cho file đó. Nhóm này đại diện cho "người ngoài" đối với file, và quyền của họ thường bị giới hạn để đảm bảo bảo mật.

Trong Linux, mỗi file hoặc thư mục đều có 3 loại quyền chính:

- **Read (r)**: quyền đọc nội dung file
- **Write (w)**: quyền chỉnh sửa file
- **Execute (x)**: quyền chạy file (đối với file thực thi)

Các quyền này được áp dụng cho 3 đối tượng: Owner, Group , Others. Nghĩa là mỗi file sẽ có 3 nhóm quyền tương ứng với 3 đối tượng trên.

Ví dụ: Một file có thể có quyền

- Owner: được đọc, ghi, chạy
- Group: chỉ được đọc
- Others: không có quyền gì

## Chmod và chown là gì ?

**Chmod** là lệnh dùng để thay đổi quyền truy cập của file hoặc thư mục đối với owner, group và others. Thông qua chmod, bạn có thể quy định:

- Ai được đọc file
- Ai được sửa file
- Ai được chạy file

Ví dụ: bạn có thể cho phép owner có đầy đủ quyền, group chỉ được đọc, và others không có quyền truy cập.

Trong Linux, quyền truy cập của file có thể được biểu diễn dưới dạng bit nhị phân (binary), trong đó mỗi quyền tương ứng với một bit:

- Read (r) = 4 = 100 (binary)
- Write (w) = 2 = 010 (binary)
- Execute (x) = 1 = 001 (binary)

Mỗi nhóm quyền (Owner, Group, Others) được biểu diễn bằng 3 bit tương ứng với ba quyền r (read), w (write), và x (execute). Mỗi quyền sẽ tương ứng với một bit: r = 1, w = 1, x = 1.

Ví dụ, nếu một nhóm có đầy đủ cả ba quyền r, w, x thì ta có dạng bit là 111 (binary). Khi chuyển sang hệ thập phân, giá trị này tương ứng với 4 + 2 + 1 = 7.

Do đó, khi muốn cấp đầy đủ quyền cho cả Owner, Group và Others, ta sử dụng lệnh:
**chmod 777**.

Lệnh này có nghĩa là cả ba nhóm Owner, Group và Others đều được cấp đầy đủ quyền đọc, ghi và thực thi trên file hoặc thư mục.

**Chown** là lệnh dùng để thay đổi người sở hữu (owner) và nhóm sở hữu (group) của file hoặc thư mục.  Điều này giúp bạn kiểm soát file thuộc về user nào và group nào trên hệ thống.

Ví dụ:

- Chuyển quyền sở hữu file sang một user khác

Gán file vào một group để nhiều người trong nhóm có thể truy cập

# Chương 2: Cài đặt Ubuntu lên server vật lý

Chương này trình bày quy trình cài đặt Ubuntu trên server vật lý, bao gồm các bước từ tải file ISO, tạo USB boot, khởi động và boot vào USB, cho đến cấu hình các tùy chọn cài đặt như ngôn ngữ, keyboard và tạo tài khoản. Nội dung giúp người đọc nắm được toàn bộ các bước cần thiết để triển khai một hệ thống Ubuntu hoàn chỉnh trên server.

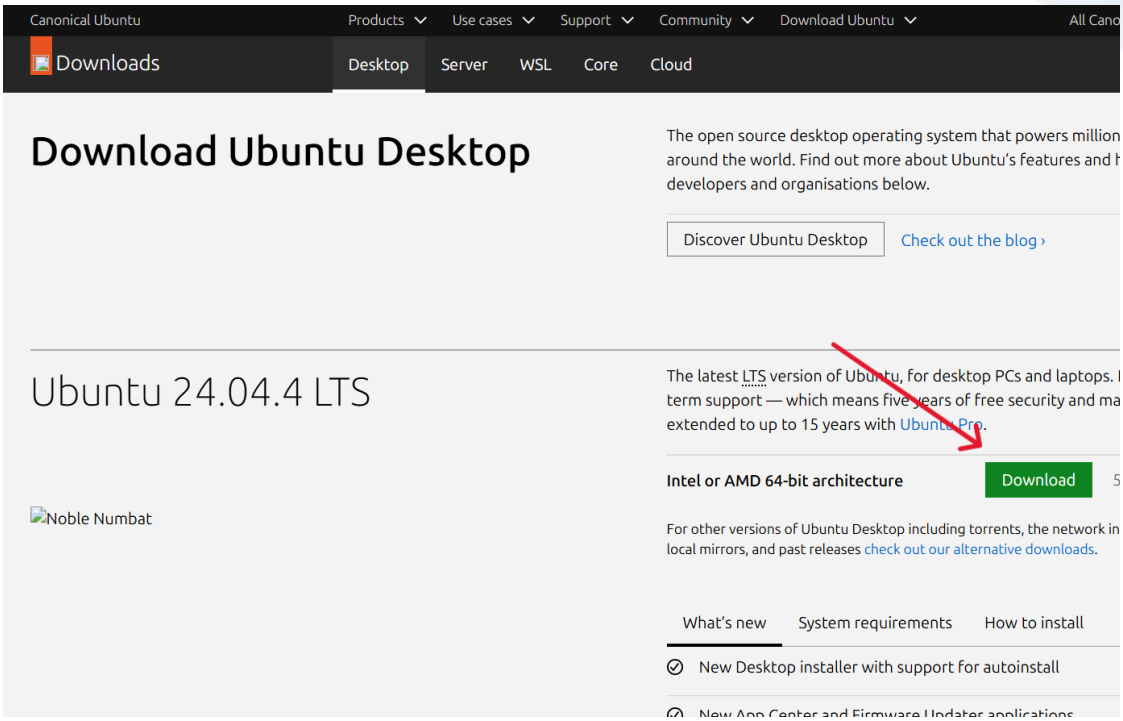
## Bắt đầu

14. Dowload file ISO
15. Burn file ISO vào USB disk
16. Khởi động server và boot vào USB disk
17. Các option cần chọn

---

## Dowload file ISO

Trước tiên, bạn cần tải file cài đặt Ubuntu (file .iso) từ trang chính thức của Ubuntu.




**Lựa chọn phiên bản**

- Ubuntu Server:
  - Không có giao diện đồ họa (GUI)
  - Chủ yếu thao tác bằng dòng lệnh
  - Tối ưu cho hệ thống server, nhẹ và ít tài nguyên
- Ubuntu Desktop:
  - Có giao diện đồ họa (GUI)
  - Cho phép thao tác trực quan bằng chuột và cửa sổ
  - Dễ sử dụng, đặc biệt khi mới làm quen hoặc muốn theo dõi hệ thống qua giao diện
  - Có các công cụ sẵn như trình duyệt, file manager, system monitor để: Theo dõi CPU, RAM, Disk,... Quản lý file trực quan và cấu hình hệ thống dễ dàng hơn.

**Trong trường hợp này, bạn chọn Ubuntu Desktop vì:**

- Có giao diện đồ họa giúp thao tác dễ hơn

- Thuận tiện theo dõi các thông số hệ thống
- Phù hợp khi bạn muốn quản lý server theo cách trực quan thay vì chỉ dùng dòng lệnh

## Burn file ISO vào USB disk

Để cài đặt Ubuntu, bạn cần tạo một USB bootable từ file ISO đã tải. USB này sẽ được dùng để khởi động và cài hệ điều hành lên server.

**Yêu cầu:**

- USB dung lượng tối thiểu 8GB
- Sao lưu dữ liệu trong USB (vì sẽ bị xóa toàn bộ)

Cách dùng Rufus để thực hiện trên Windows

***Bước 1: Tải Rufus***

- Truy cập trang chính thức của Rufus
- Tải phiên bản mới nhất (không cần cài đặt, chạy trực tiếp file .exe)

- bạn cần phải tạo USB chứa trình cài đặt từ ISO có khả năng khởi động (Windows,
- bạn cần làm việc trên hệ thống chưa cài đặt HĐH
- bạn cần nạp BIOS hoặc phần mềm lõi khác từ DOS
- bạn muốn chạy tiện ích cấp thấp

Mặc dù có kích thước nhỏ, Rufus có thể cung cấp mọi thứ bạn cần!

Một danh sách chưa đầy đủ của các tệp ISO được hỗ trợ cũng được cung cấp ở cuối t

## Tải xuống

**Phiên bản mới nhất:**

| ink | Loại | Nền tảng | Kích thước | Ngà |
|---|---|---|---|---|
| rufus-4.13.exe | Chuẩn | Windows x64 | 1.9 MB | 202 |
| rufus-4.13p.exe | Phiên bản di động | Windows x64 | 1.9 MB | 202 |
| rufus-4.13_x86.exe | Chuẩn | Windows x86 | 1.8 MB | 202 |
| rufus-4.13_arm64.exe | Chuẩn | Windows ARM64 | 4.9 MB | 202 |

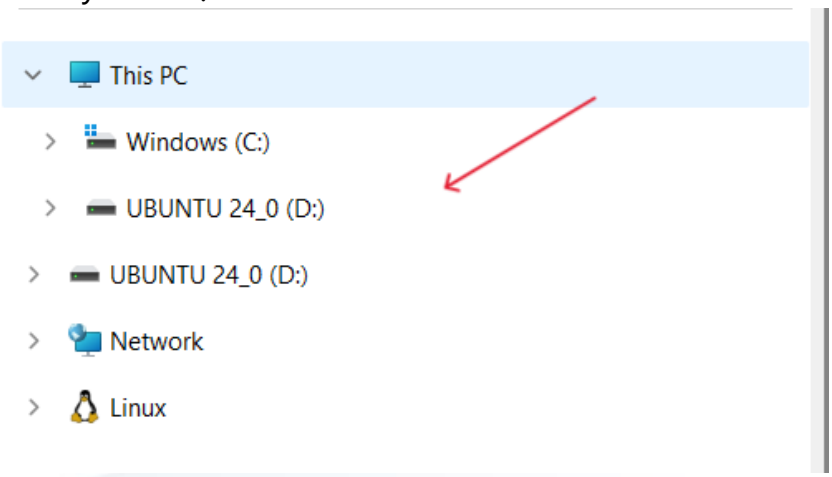
Phiên bản khác (GitHub)
Phiên bản khác (dAppCDN)

**Yêu cầu hệ thống:**

Windows 8 hoặc mới hơn. Chỉ cần một lần tải xuống là ứng dụng đã sẵn sàng để sử d

### *Bước 2: Cắm USB vào máy*

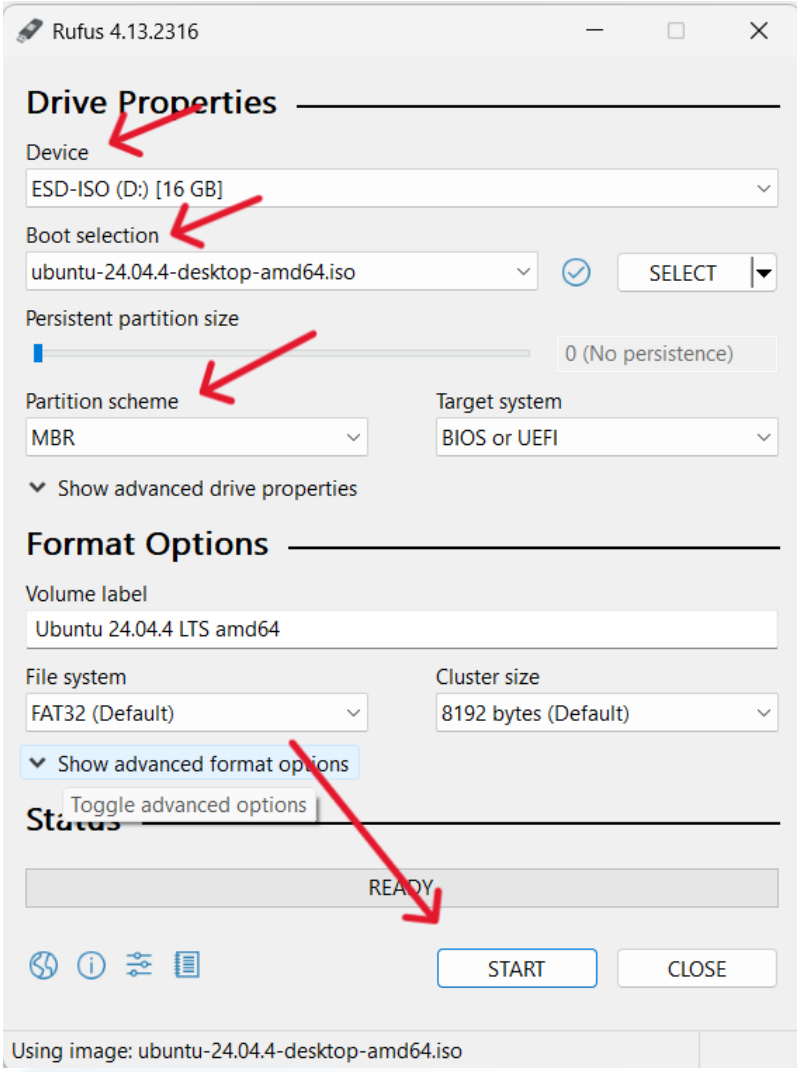
- Cắm USB vào máy tính
- Đảm bảo máy đã nhận USB



***Bước 3: Mở Rufus và cấu hình***

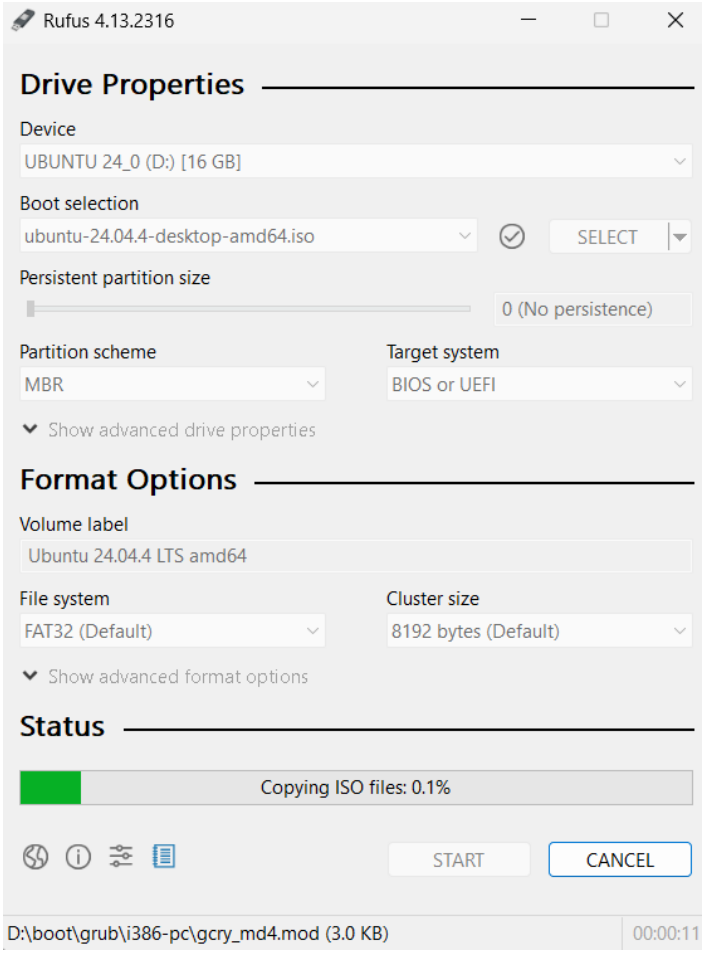
Khi mở Rufus, bạn sẽ thấy giao diện với các mục cấu hình:

- Device
  - Chọn đúng USB bạn vừa cắm
- Boot selection
  - Nhấn SELECT
  - Chọn file ISO Ubuntu đã tải (ví dụ: ubuntu-22.04-desktop-amd64.iso)
- Partition scheme: Chọn MBR (dùng cho BIOS Legacy).GPT không dùng được trong trường hợp này.

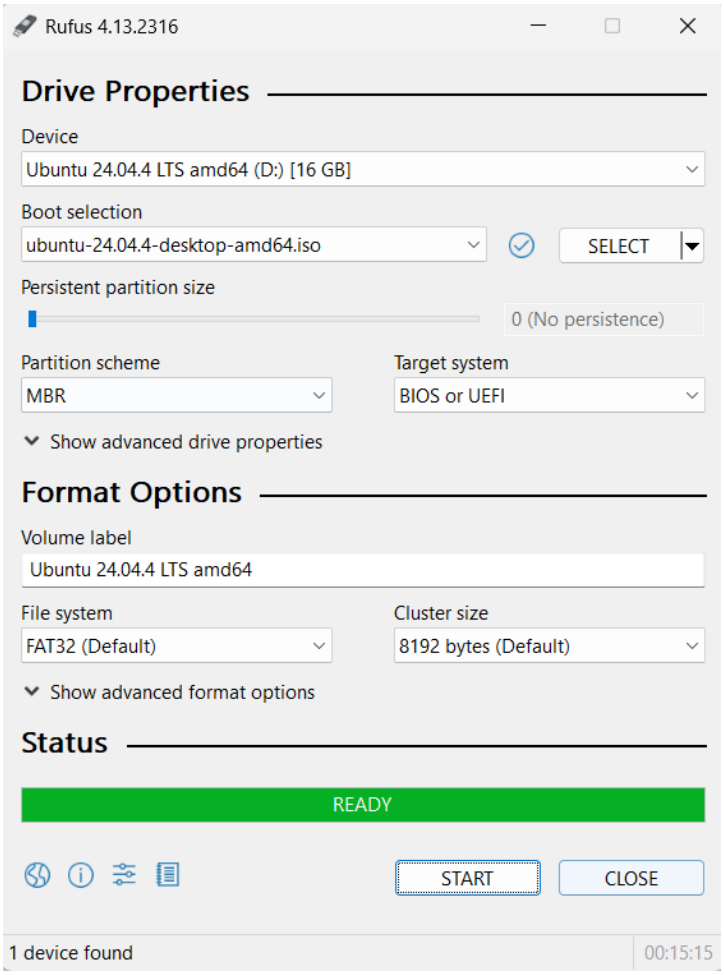


***Bước 4: Bắt đầu tạo USB boot***

- Nhấn Start
- Rufus sẽ cảnh báo dữ liệu trong USB sẽ bị xóa → chọn OK
- Chờ quá trình hoàn tất (vài phút)




***Bước 5: Sau khi quá trình burn dữ liệu hoàn tất, bạn sẽ thấy màn hình sau:***




# Khởi động server và boot vào USB disk

Sau khi đã tạo USB boot, bạn tiến hành khởi động server và chọn boot từ USB để bắt đầu quá trình cài đặt Ubuntu.

**Hướng dẫn thực hiện**

***Bước 1: Cắm USB và khởi động máy***

- Cắm USB boot vào server
- Bật nguồn hoặc khởi động lại máy

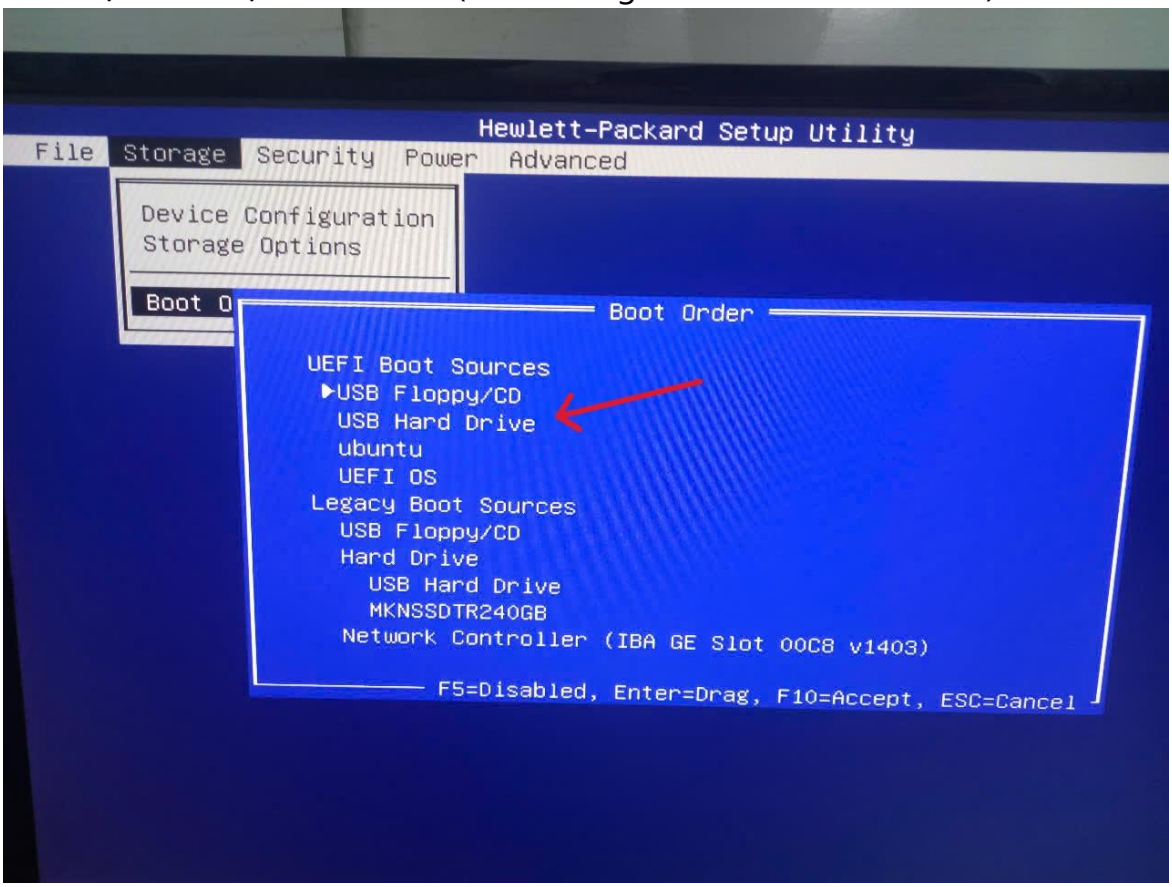
***Bước 2: Truy cập Boot Menu***

- Ngay khi máy vừa khởi động, bạn cần nhấn phím để vào Boot Menu (tùy từng dòng máy). Thường gặp: F2 / F10 / F12 / ESC / DEL.

- Nhấn liên tục phím này ngay khi bật máy (trước khi hệ điều hành khởi động)

Sau đó:

- Một menu sẽ hiện ra
- Chọn thiết bị boot là USB (USB Storage / USB HDD / tên USB)




*Lưu ý: Thứ tự bạn thấy trên màn hình ( USB Floppy/CD, USB Hard Drive, ubuntu và UEFI OS) là thứ tự BIOS ưu tiên tải hệ điều hành vào server. Vì USB Floppy/CD không có nên BIOS sẽ boot hệ điều hành từ UBS Hard Drive mà bạn đang cắm vào server.*

***Bước 3: Lưu và khởi động lại***

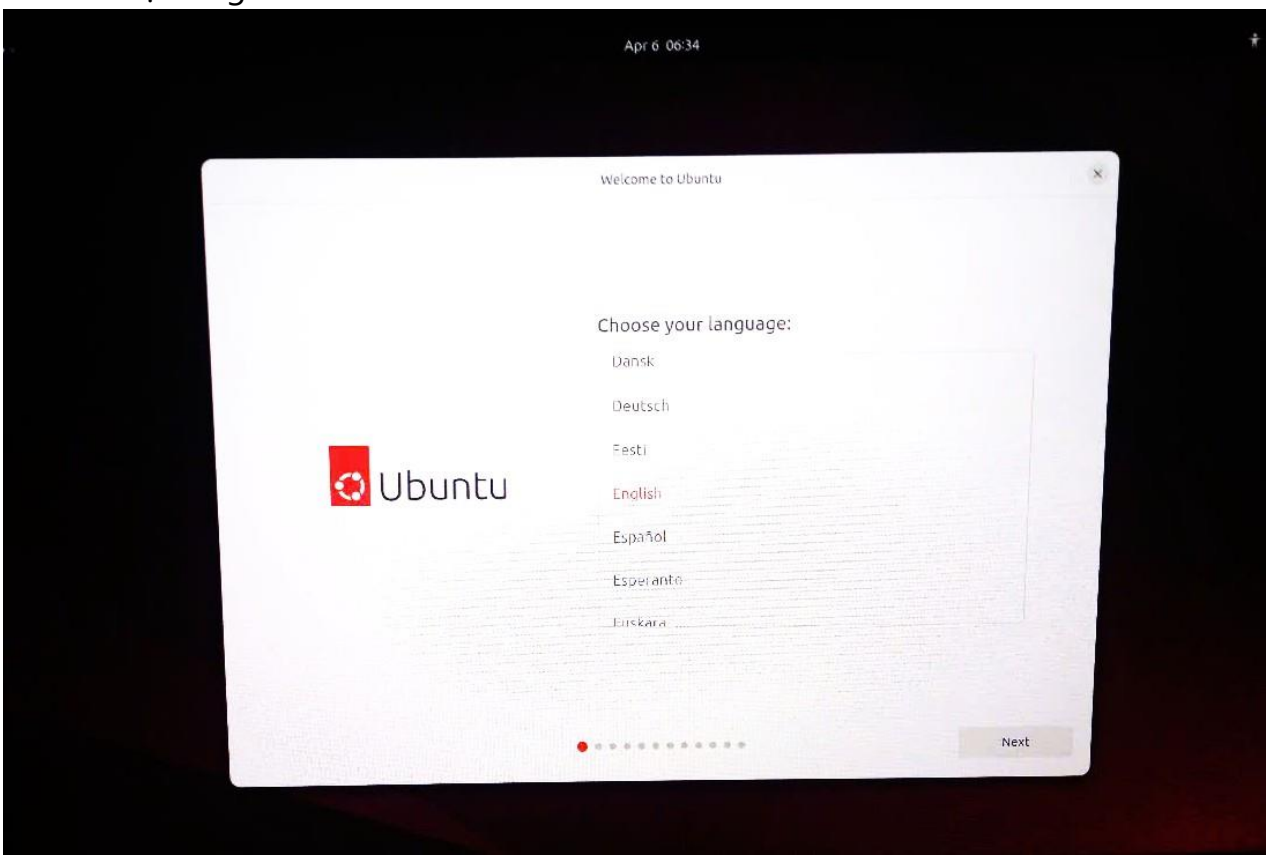
- Nhấn Save và Exit
- Máy sẽ restart

# Các option cần chọn

Sau khi boot vào USB, hệ thống sẽ hiển thị trình cài đặt Ubuntu. Bạn cần thực hiện lần lượt các bước cấu hình sau:

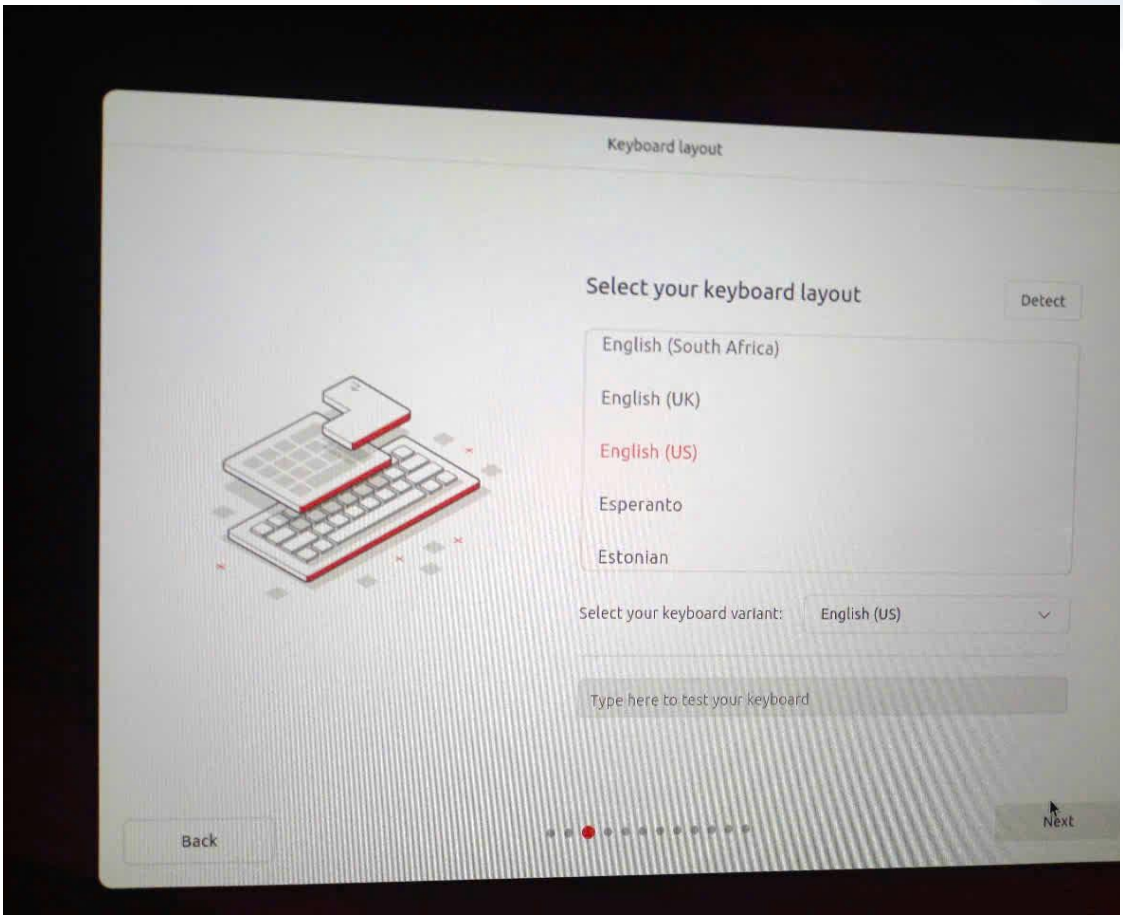
***Bước 1: Chọn ngôn ngữ***

- Chọn English




***Bước 2: Keyboard layout***

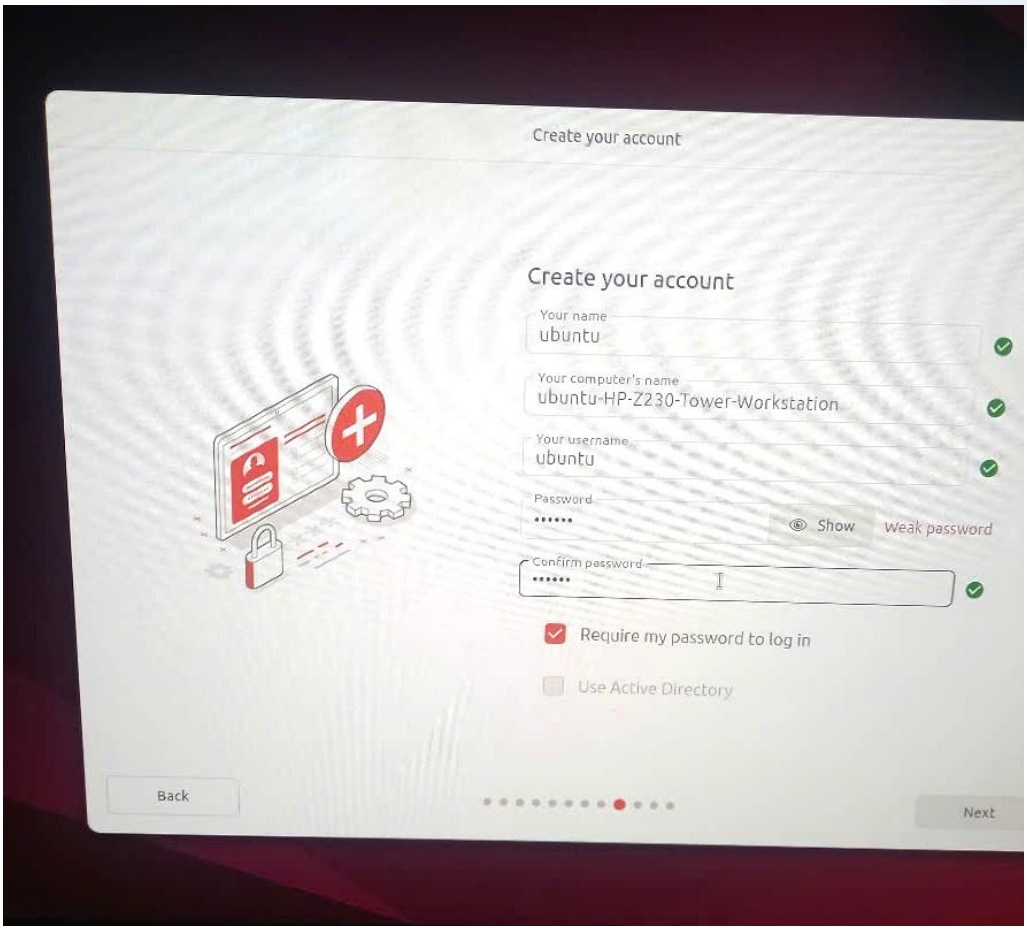
- Chọn English hoặc layout phù hợp với bàn phím bạn đang sử dụng




***Bước 3: Profile setup***

Bạn cần tạo tài khoản hệ thống:

- Name: tên hiển thị
- Server name: tên máy trong mạng (Ví dụ: Z230)
- Username: tài khoản đăng nhập
- Password: mật khẩu



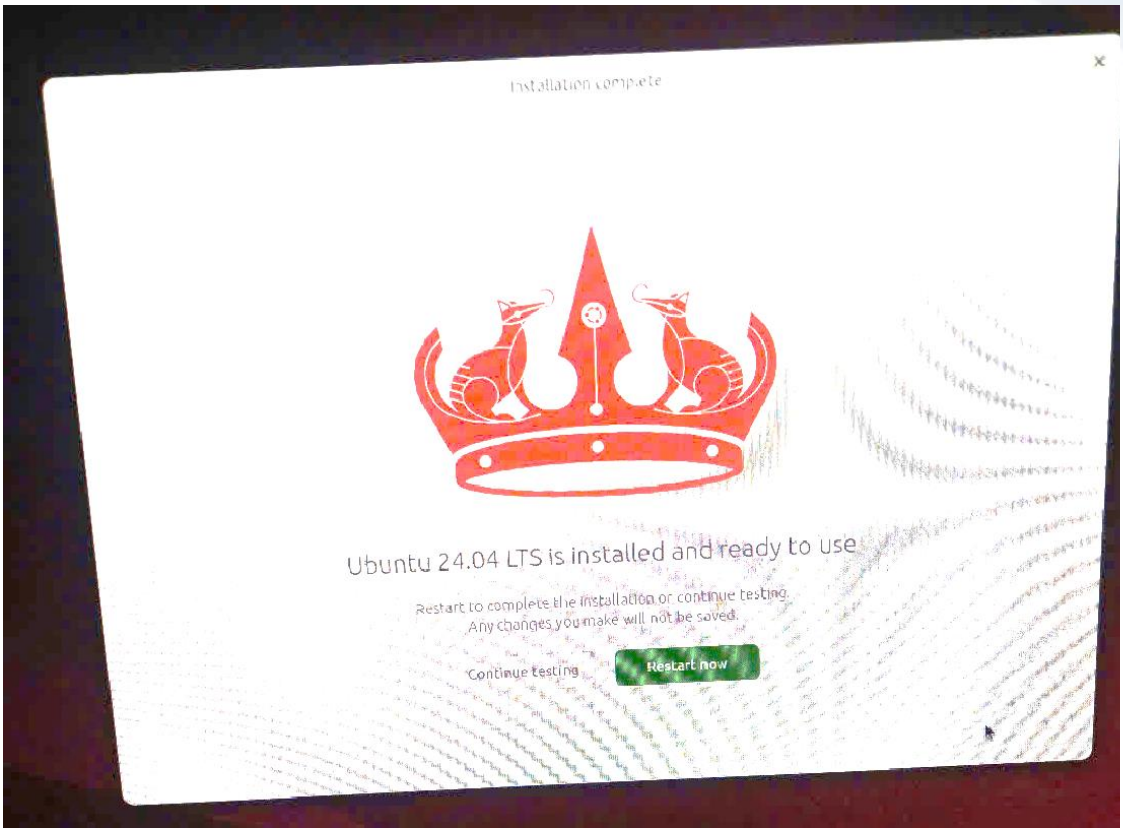

***Bước 4: Hoàn tất cài đặt***

- Hệ thống sẽ tiến hành cài đặt Ubuntu
- Quá trình này có thể mất vài phút

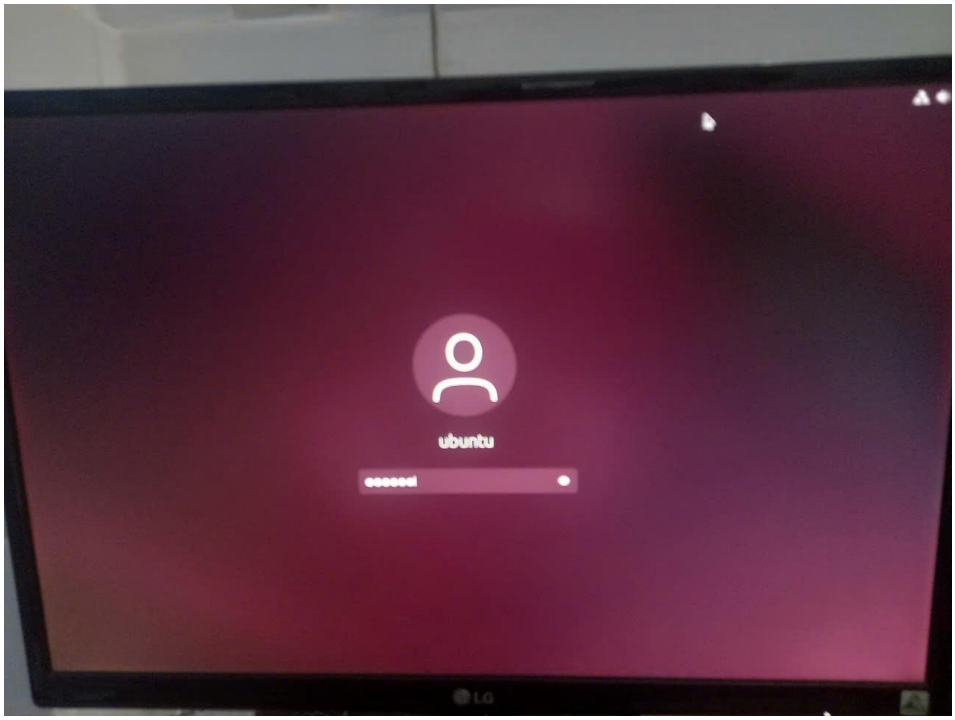
Sau khi hoàn tất:

- Chọn restart
- Rút USB ra khi máy reboot

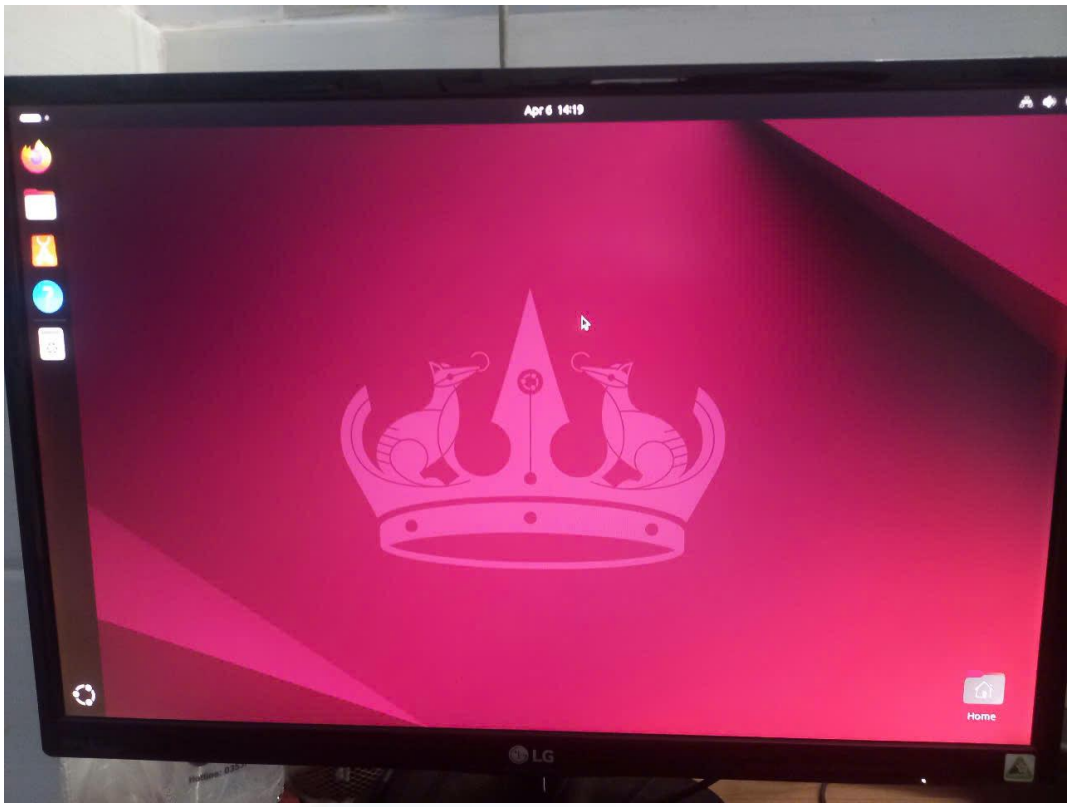
*Lưu ý: Nếu không rút USB, máy có thể boot lại vào trình cài đặt thay vì hệ điều hành vừa cài*




***Bước 5: Sau khi restart, bạn dùng username và password đã cấu hình để truy cập vào máy***




Sau khi cài đặt hoàn tất, bạn sẽ nhìn thấy giao diện của ubuntu



# Chương 3: Công cụ cần thiết trên Windows

Chương này giới thiệu các công cụ cần thiết trên Windows phục vụ cho việc kết nối và làm việc với server từ xa, bao gồm SSH, quản lý file và tạo khóa xác thực. Các công cụ này giúp người dùng thiết lập kết nối, truyền tải dữ liệu và thao tác với server một cách thuận tiện, an toàn và hiệu quả.

Các công cụ này được sử dụng thường xuyên trong thực tế, có nhiều ưu điểm so với các công cụ khác.

## Nội dung

18. **PuTTY**: tạo SSH key pair
19. **Filezilla**: dùng để quản lý và transfer file
20. **Tabby terminal**: dùng để SSH vào server

---

## PuTTY: tạo SSH key pair

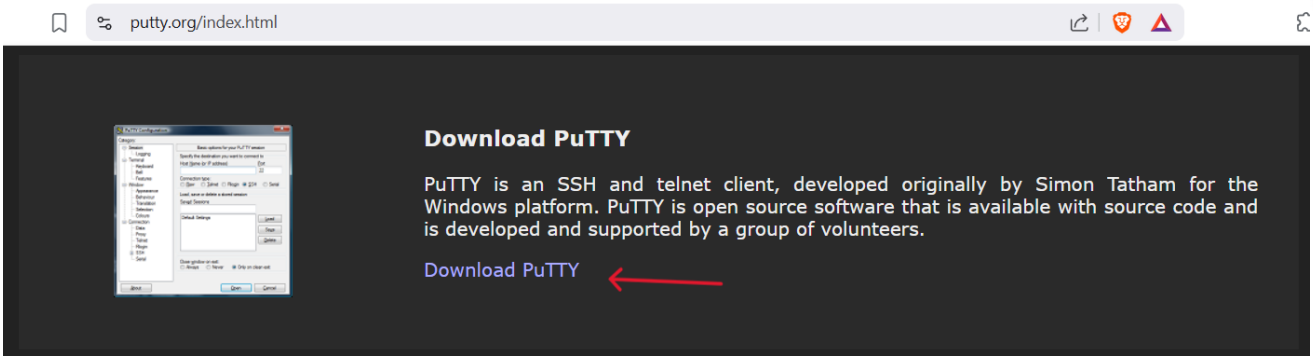
PuTTY là công cụ phổ biến trên Windows dùng để tạo và quản lý SSH key pair phục vụ cho việc xác thực khi kết nối đến server. SSH key pair bao gồm một cặp khóa: private key (khóa bí mật, cần được giữ bí mật không cho người khác biết) và public key (khóa công khai, được đưa lên server hoặc chia sẻ với mọi người).

Khi sử dụng PuTTYgen (công cụ đi kèm PuTTY), người dùng có thể tạo SSH key pair bằng cách chọn loại key (thường là RSA hoặc Ed25519), sau đó tiến hành tạo key và lưu lại cặp private key/public key dưới định dạng .ppk/.pub. Public key sẽ được sao chép và thêm vào file authorized_keys trên server.

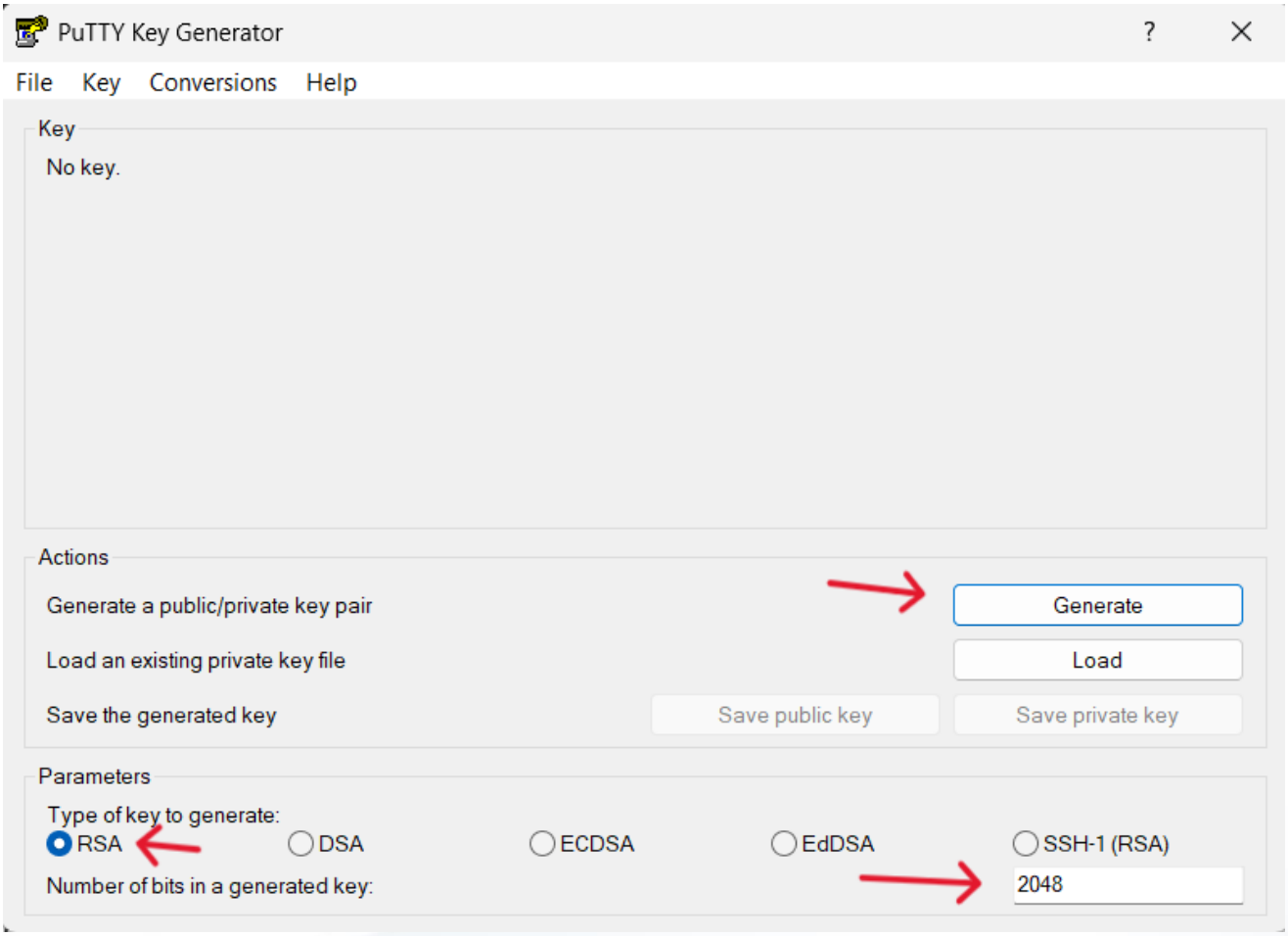
Việc sử dụng SSH key pair giúp tăng cường bảo mật, thay thế cho việc đăng nhập bằng mật khẩu truyền thống, đồng thời giúp quá trình đăng nhập vào server trở nên an toàn hơn.

**Hướng dẫn sử dụng PuTTY:**

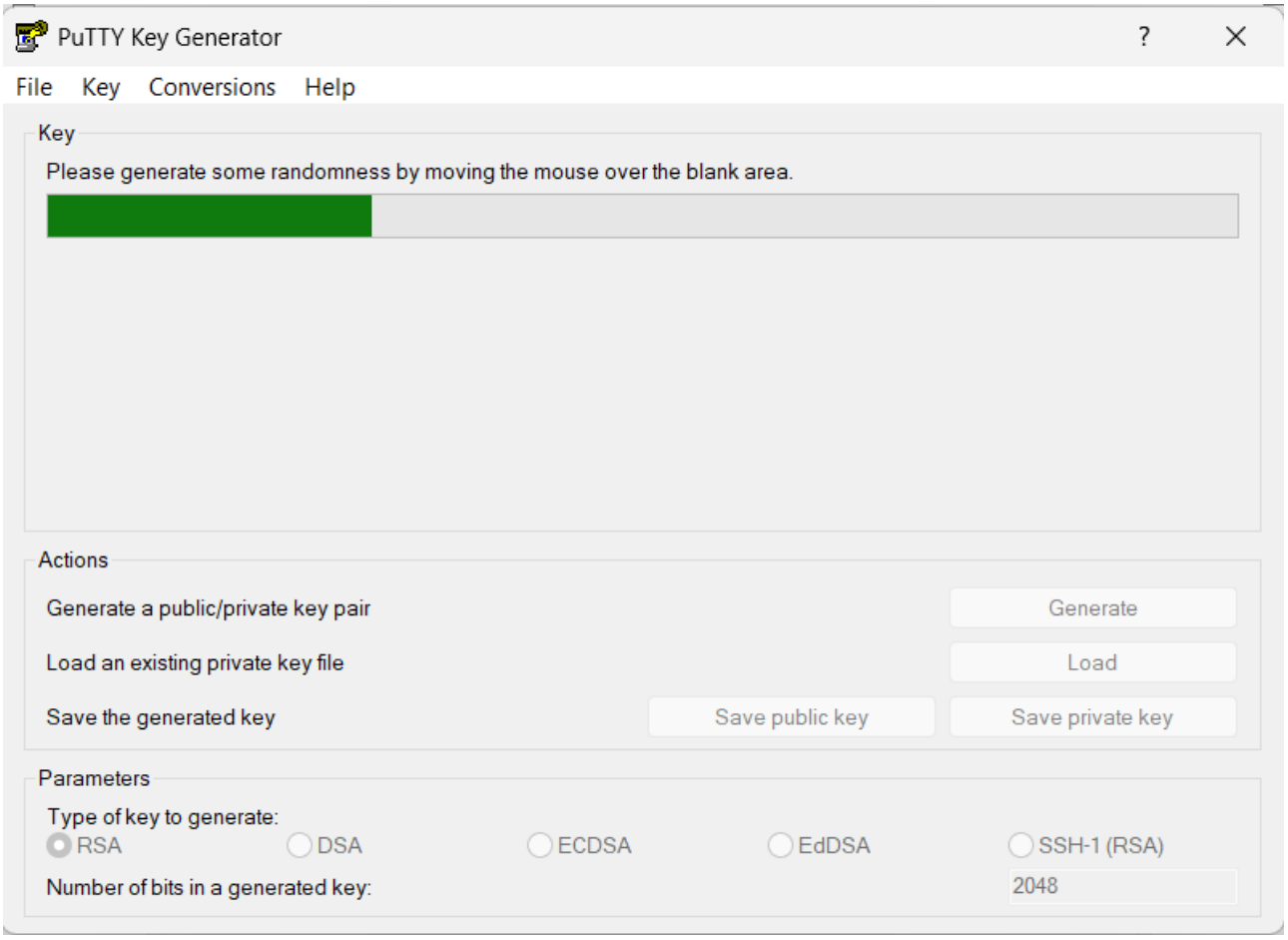
**Bước 1:** Download PuTTY từ trang chủ và cài đặt



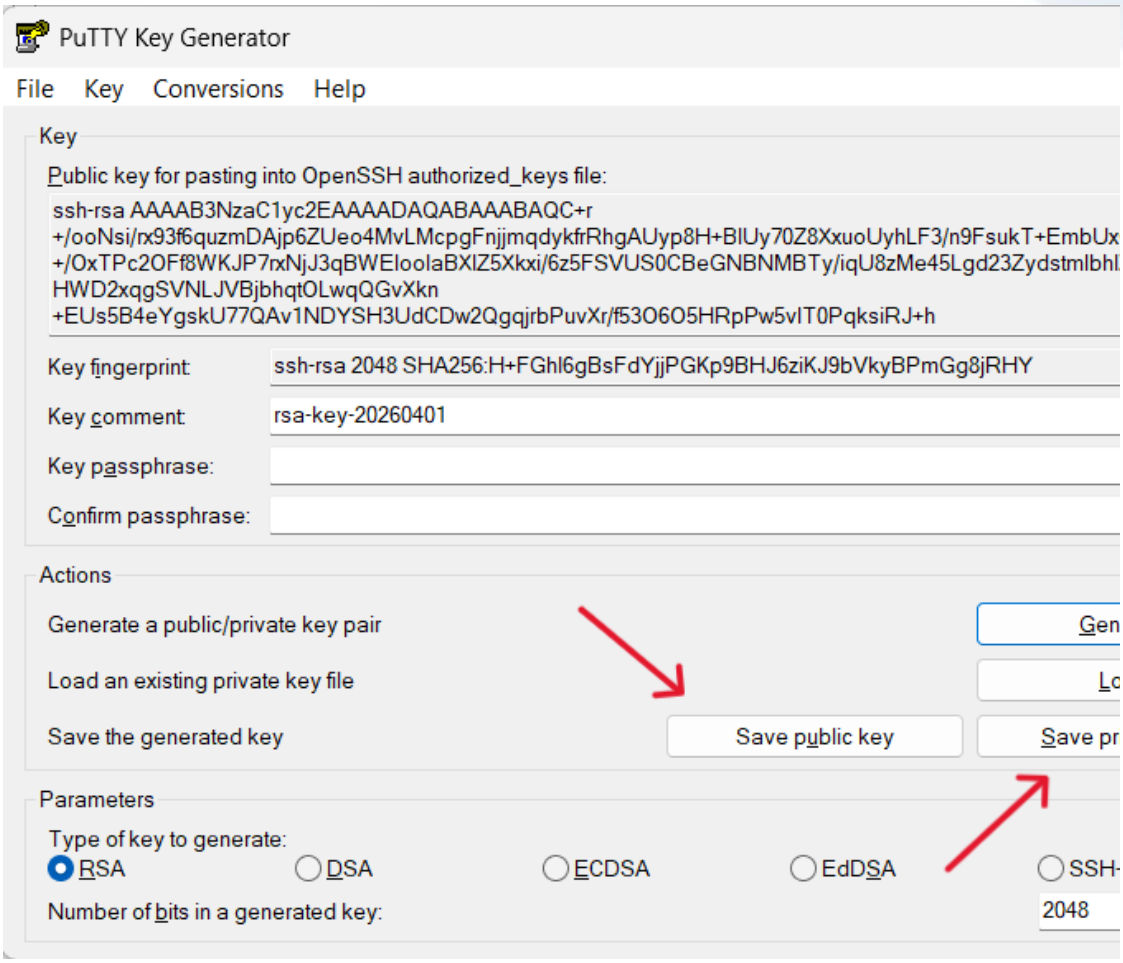

**Bước 2:** Mở PuTTYgen (công cụ đi kèm PuTTY dùng để tạo key pair). Chọn thuật toán để khởi tạo key pair là RSA và chiều dài bit là 2048.



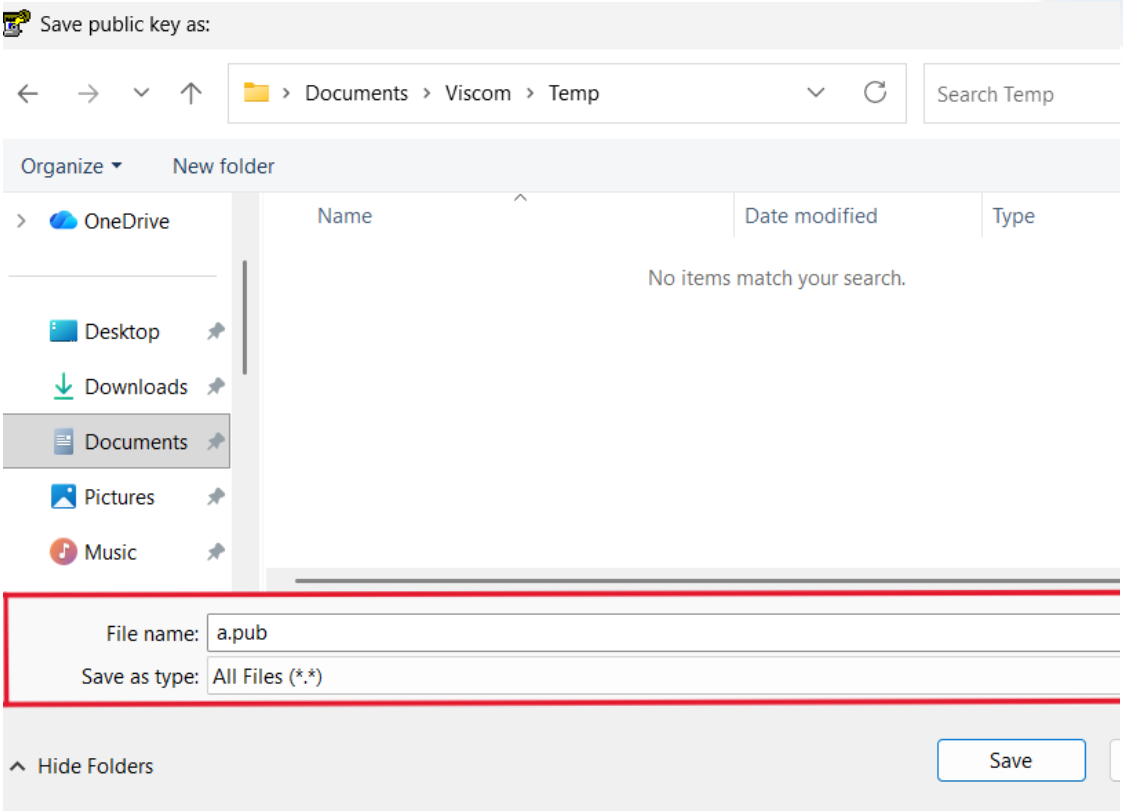
**Bước 3:** Ấn nút Generate và di chuyển chuột liên tục trong phạm vi window của PuTTY Key Generator ( lưu ý: chỉ khi nào bạn di chuyển chuột trong cửa sổ PuTTY thì thanh quá trình màu xanh mới hoạt động).



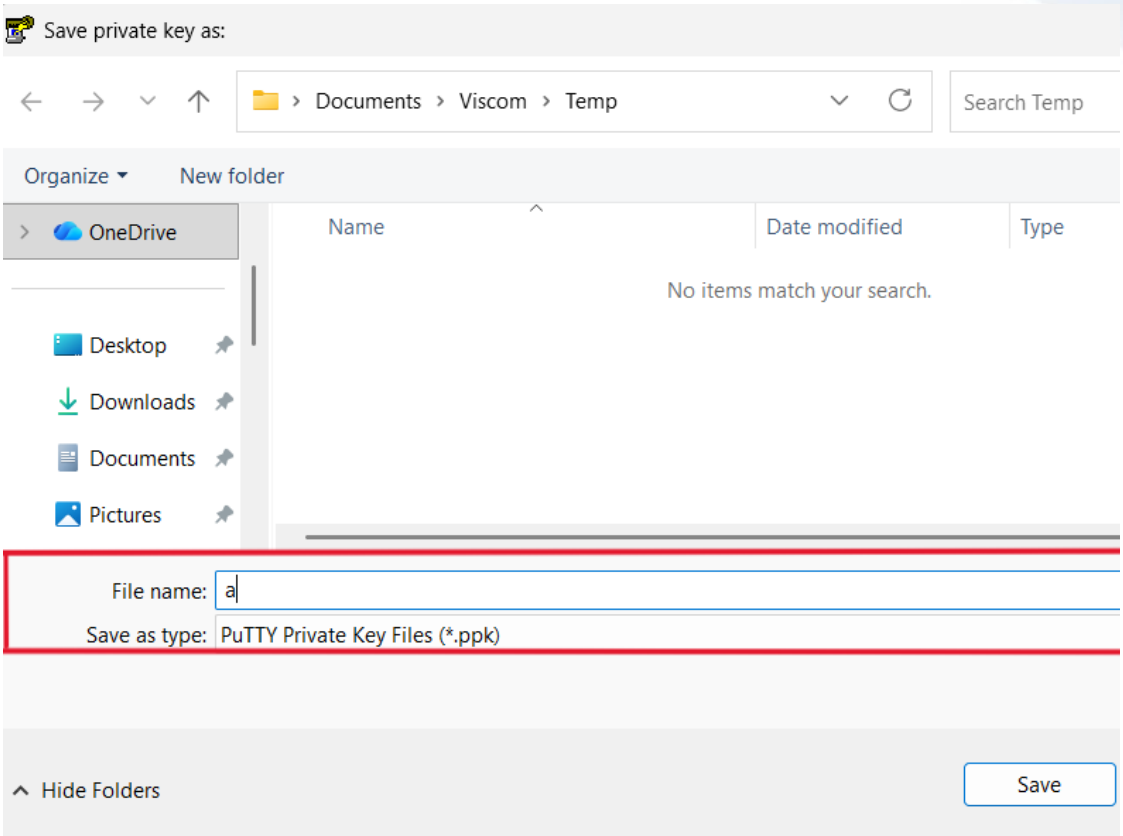

**Bước 4:** Sau khi hoàn tất quá trình, bạn sẽ nhận được key pair như hình (public key và private key). Chọn nút “save public key” để lưu public key.



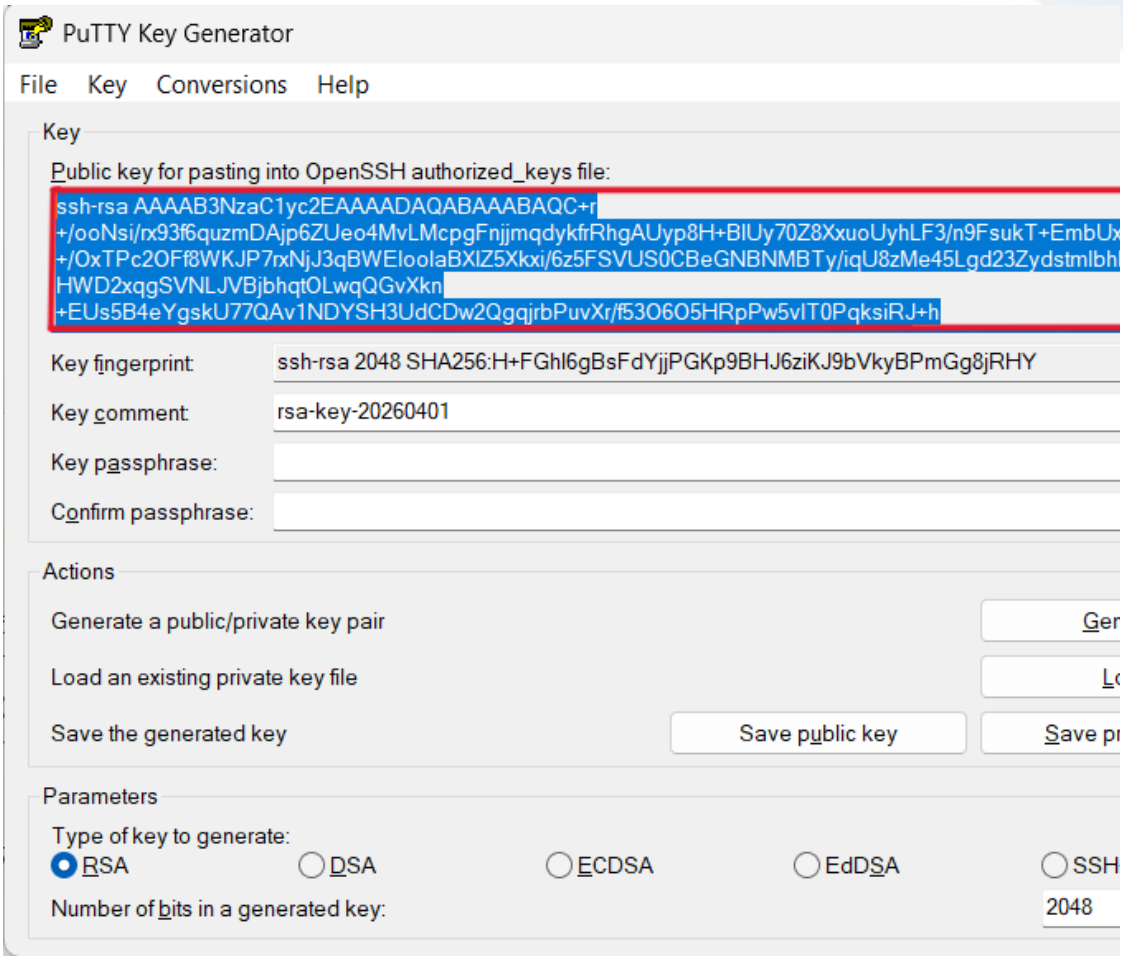

**Bước 5:** Lưu public key với đuôi ".pub" trong thư mục bạn mong muốn



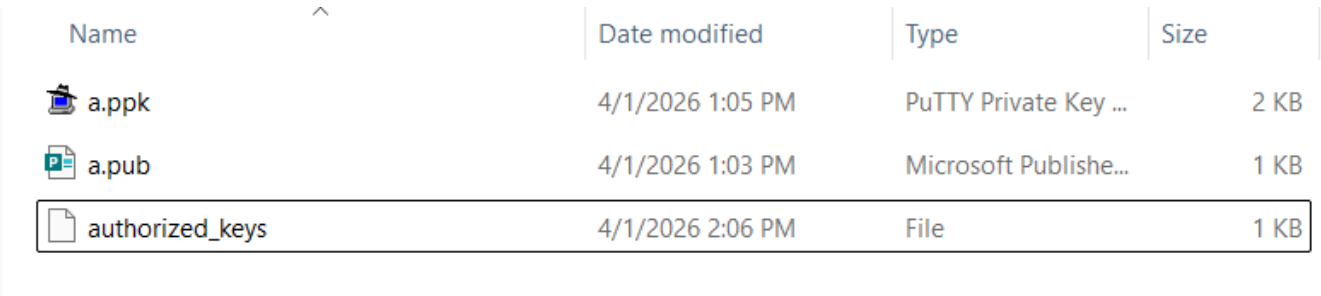

**Bước 6:** Chọn “save private key” và lưu với đuôi “.ppk”




**Bước 7:** Copy phần key được bôi vào file “authorized_keys” để tiện đưa lên server




**Bước 8:** Kiểm tra lại key pair đã khởi tạo trong thư mục



| Name | Date modified | Type | Size |
|---|---|---|---|
| a.ppk | 4/1/2026 1:05 PM | PuTTY Private Key ... | 2 KB |
| a.pub | 4/1/2026 1:03 PM | Microsoft Publishe... | 1 KB |
| authorized_keys | 4/1/2026 2:06 PM | File | 1 KB |

**Lưu ý:** Private key (khóa riêng) là thông tin bảo mật quan trọng, cần được lưu trữ cẩn thận và tuyệt đối không chia sẻ, vì nó được sử dụng để xác thực khi kết nối đến server. Ngược lại, Public key (khóa công khai) có thể được chia sẻ và được cài đặt trên server để cho phép các kết nối hợp lệ từ phía người dùng.

# FileZilla: dùng để quản lý và transfer file

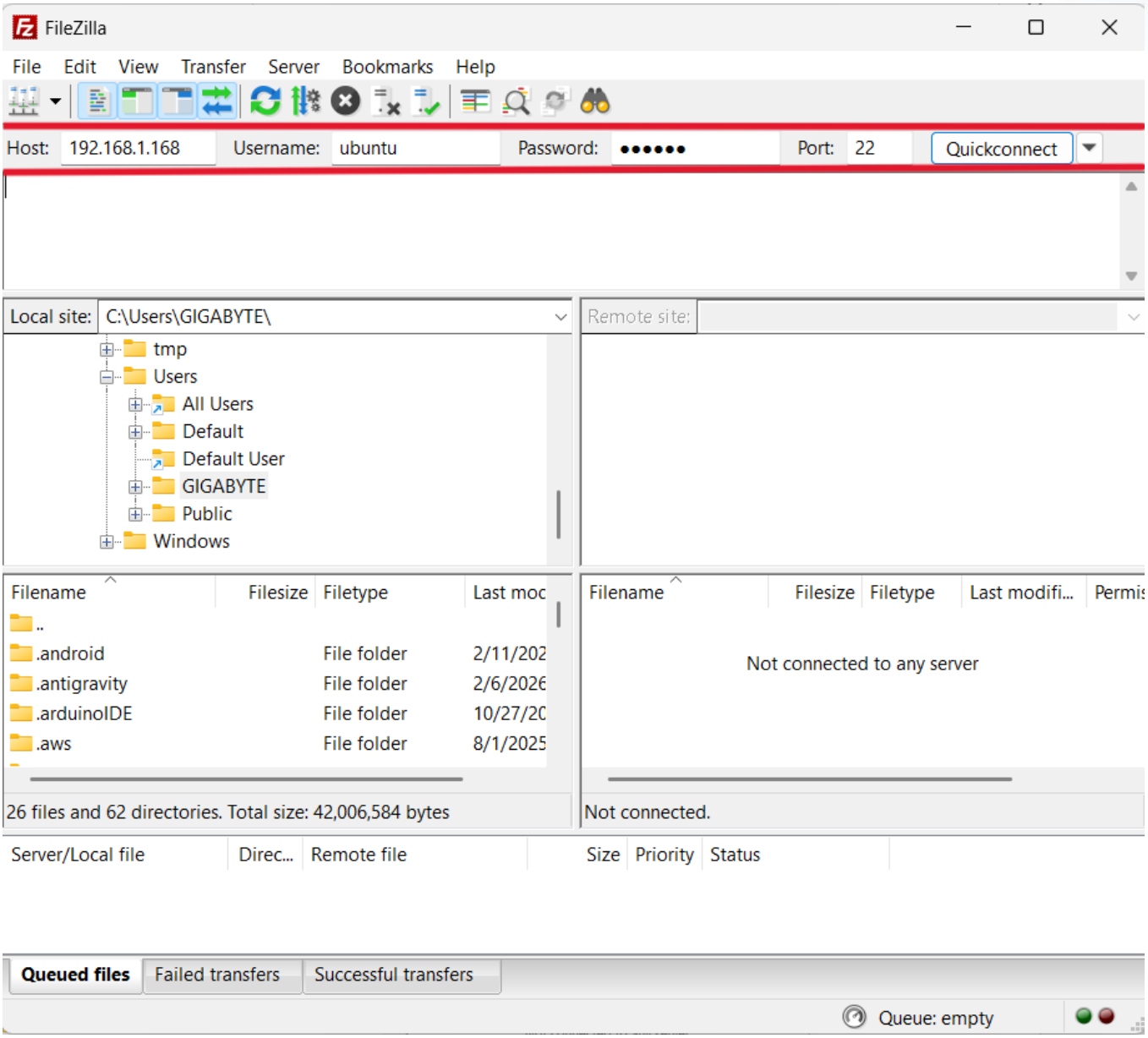
FileZilla là công cụ giúp bạn gửi và nhận file giữa máy tính và server. Bạn có thể upload (đưa file lên server) hoặc download (tải file về máy) một cách dễ dàng. Ngoài ra, FileZilla còn giúp bạn xem và quản lý các thư mục trên server giống như dùng File Explorer trên Windows, nên rất dễ sử dụng, không cần dùng lệnh phức tạp.

**Hướng dẫn sử dụng FileZilla:**

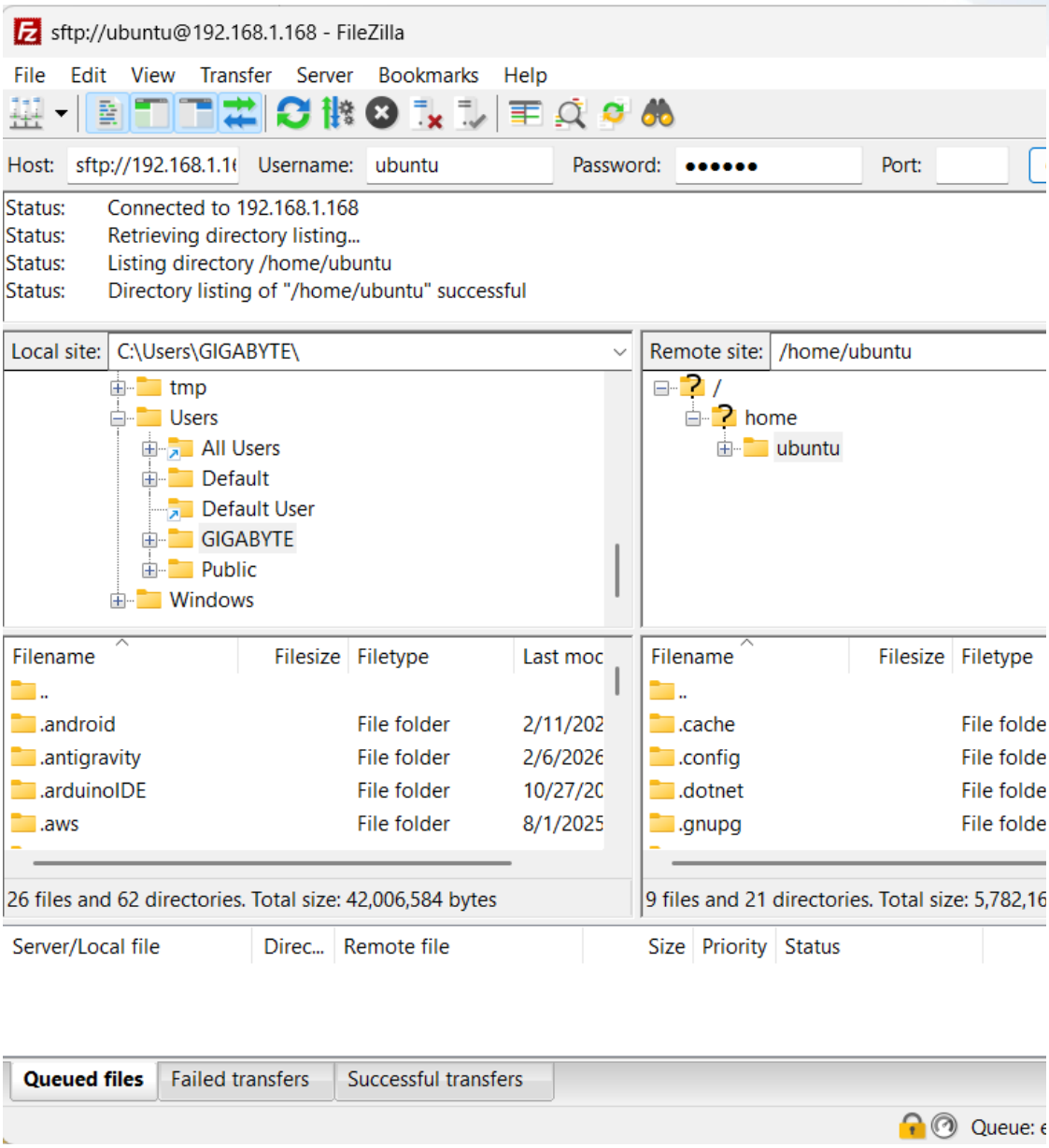
**Bước 1:** Download FileZilla từ trang chủ và cài đặt

**Bước 2:** Mở công cụ FileZilla và điền các thông tin trong ô màu đỏ để kết nối với server.

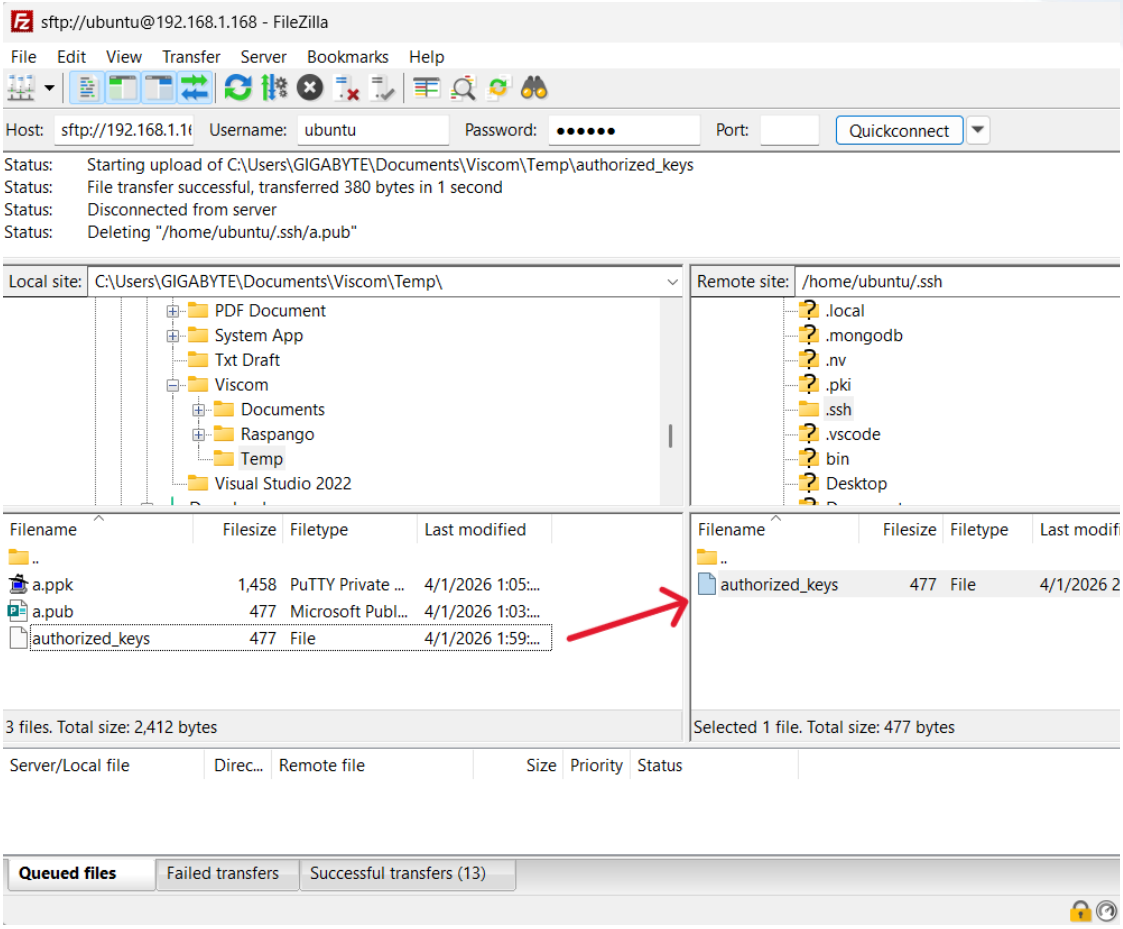
**Lưu ý:** Khi chưa có phương thức xác thực bằng keypair, bạn tạm sử dụng username và password của server được cung cấp.




**Bước 3:** Sau khi kết nối thành công đến server, giao diện FileZilla sẽ hiển thị toàn bộ thư mục và file trên server. Từ đây, bạn có thể thao tác với các file (copy, xóa, upload, download,...) một cách dễ dàng, tương tự như khi làm việc trên Windows (Những ô ở bên trái là thư mục trên máy người dùng, ô bên phải là thư mục trên server).




**Bước 4:** Copy Public key bạn vừa tạo ở trên vào folder .ssh trên server bằng cách kéo thả file authorized_keys



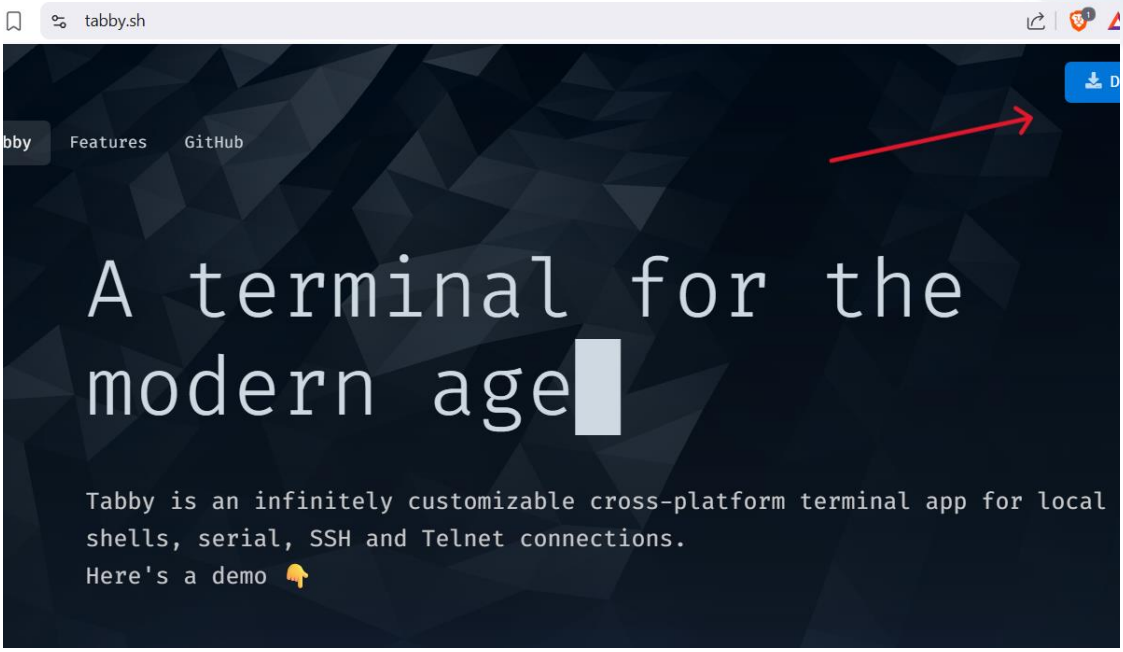

## Tabby terminal: dùng để SSH vào server

Tabby Terminal là công cụ dùng để kết nối đến server thông qua SSH. Sau khi nhập thông tin như địa chỉ IP, username và SSH key (hoặc mật khẩu), bạn có thể truy cập và thực hiện các lệnh trực tiếp trên server.

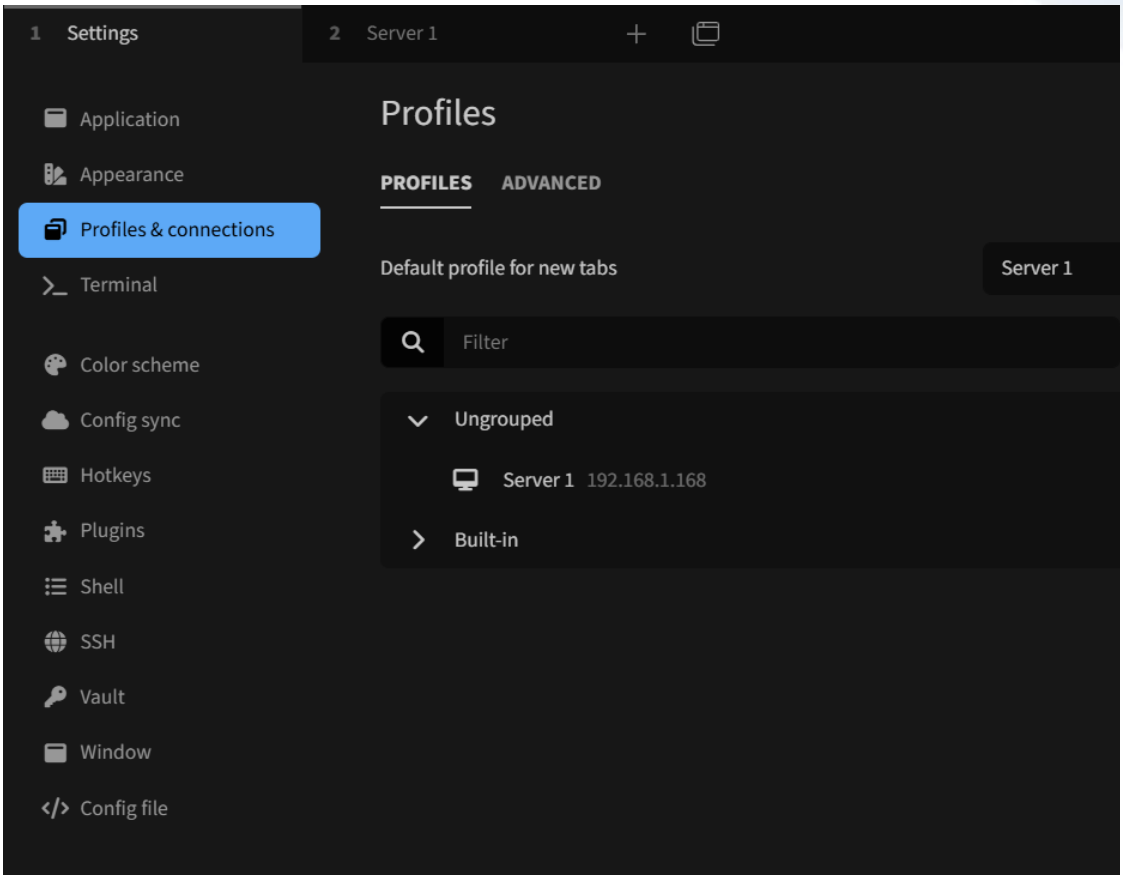
So với việc dùng CMD thông thường, Tabby có nhiều ưu điểm như: lưu lại lịch sử kết nối để lần sau truy cập nhanh hơn, quản lý nhiều phiên làm việc cùng lúc theo dạng tab, giao diện dễ nhìn và hỗ trợ cấu hình linh hoạt. Nhờ đó, việc kết nối và làm việc với server trở nên nhanh chóng và thuận tiện hơn.

**Hướng dẫn sử dụng:**

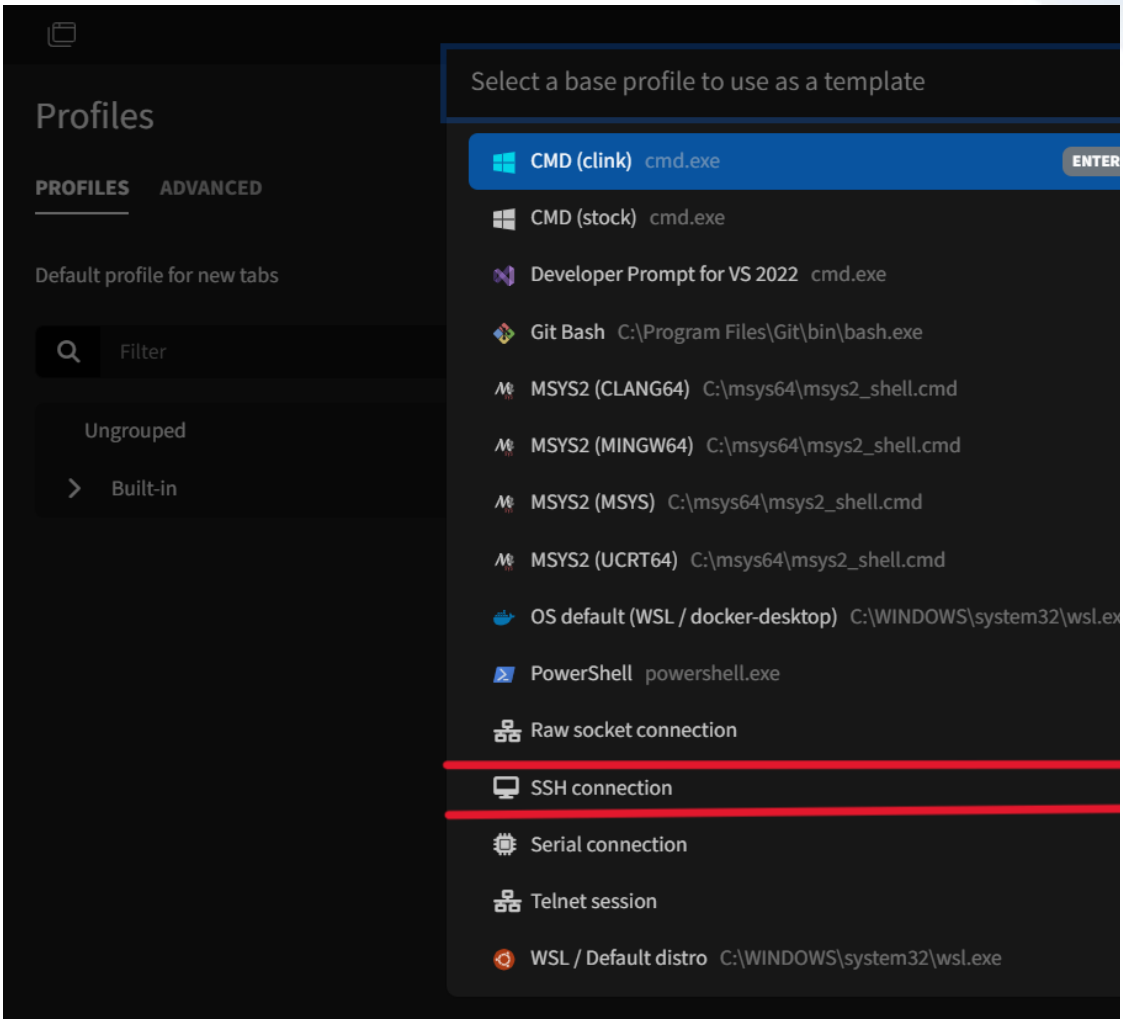
**Bước 1:** Download Tabby terminal từ trang chủ và cài đặt




**Bước 2:** Sau khi truy cập Tabby terminal, bạn sẽ thấy giao diện như hình. Thanh công cụ bên trái dùng để cấu hình các chức năng của Tabby. Profile dùng để lưu lại các cấu hình truy cập trước vào các server (có thể sử dụng lại để tiết kiệm thời gian).



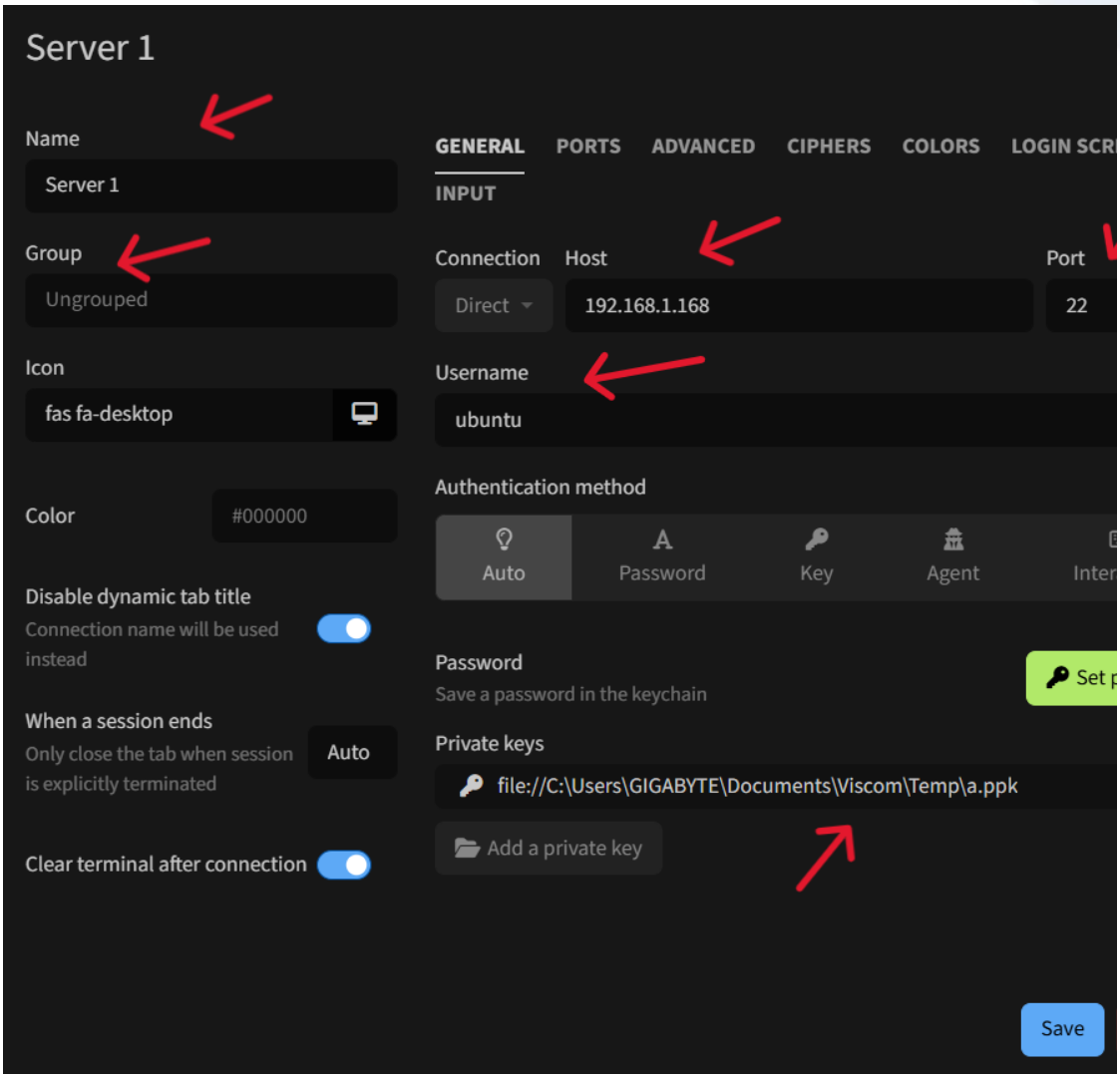

**Bước 3:** Chọn nút New để tạo kết nối mới. Sau đó chọn loại kết nối là SSH, để kết nối đến server.




**Bước 4:** Cấu hình các trường thông tin sau để kết nối với server. ( Những chỗ màu đỏ là những thông tin cần phải điền). Trong đó:

- **Name** là tên cho profile truy cập
- **Group** là nhóm chứa profile này
- **Host** là địa chỉ IP của server
- **Port** là cổng kết nối SSH, mặc định là 22 (nhập đúng cổng server đang dùng).
- **Username** là tên user trên server để đăng nhập (ví dụ: ubuntu, root).
- **Private key** là khóa bí mật mà bạn đã tạo ở trên.

Lưu ý: Vì ở trên đã copy public key lên server rồi nên giờ tăng dùng key để xác thực (không dùng password)




**Bước 5:** Sau khi kết nối thành công, giao diện tabby sẽ hiện ra terminal của server. Bạn dùng terminal này để tương tác với server

```
1 Settings        2 Server 1

ubuntu@192.168.1.168:22

Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

178 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Mon Dec 22 17:25:48 2025 from 192.168.1.72
ubuntu@Z230:~$
```

# Chương 4: Cấu hình và cài đặt phần mềm cần thiết

Chương này trình bày các bước cài đặt và cấu hình các công cụ quan trọng như OpenSSH, Git, Nginx và Certbot, đồng thời thiết lập các biện pháp bảo mật như tắt đăng nhập bằng mật khẩu và bật firewall. Các nội dung này nhằm đảm bảo hệ thống hoạt động ổn định, an toàn và sẵn sàng cho việc triển khai thực tế.

## Nội dung


21. Cài đặt OpenSSH
22. Tắt chức năng đăng nhập bằng password
23. Bật firewall
24. Cài đặt git
25. Cài đặt nginx


---

## Cài đặt OpenSSH

OpenSSH là bộ công cụ dùng để kết nối và quản lý server từ xa thông qua giao thức SSH.

**Hướng dẫn cài đặt:**

***Bước 1: Cập nhật và cài đặt OpenSSH. Sử dụng lệnh:***

```
sudo apt update

sudo apt install openssh-server -y
```

***Bước 2: Kiểm tra dịch vụ***

```
sudo systemctl status ssh
```

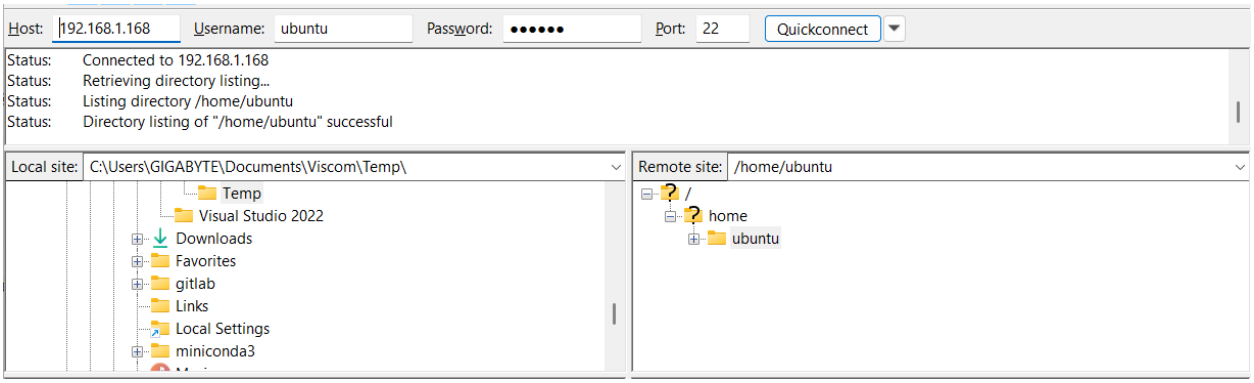
***Bước 3: Khởi động dịch vụ SSH ngay lập tức***

```
sudo systemctl start ssh
```

***Bước 4: Tự động khởi động SSH mỗi khi máy boot***

```
sudo systemctl enable ssh
```

***Bước 5: Kiểm tra kết nối từ máy khác bằng FileZilla đã được hướng dẫn:***

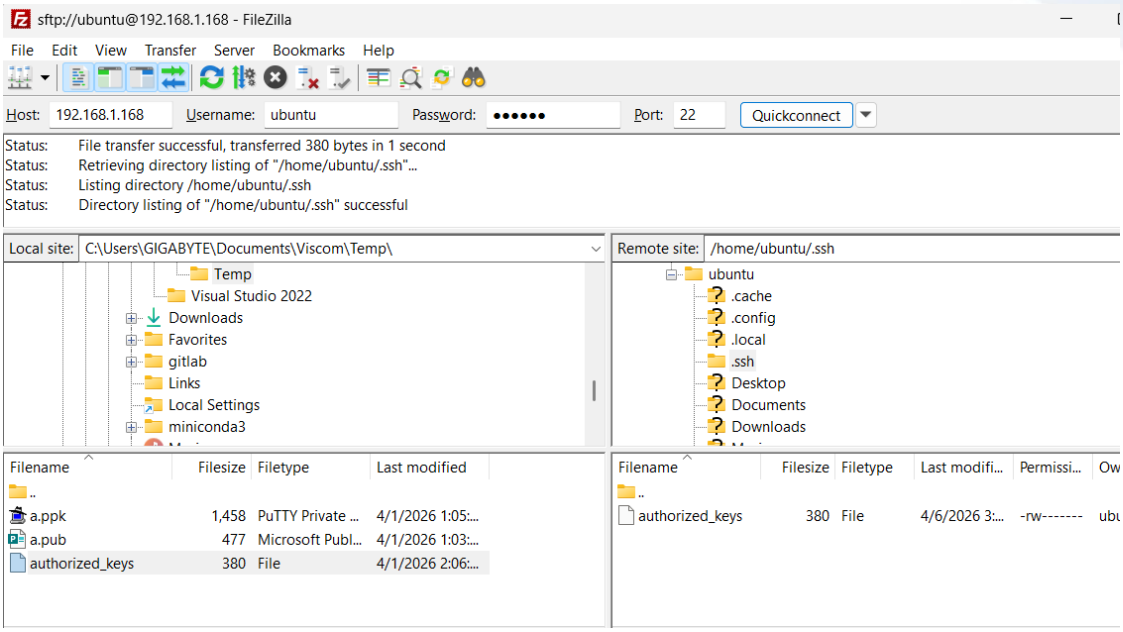



# Tắt chức năng đăng nhập bằng password

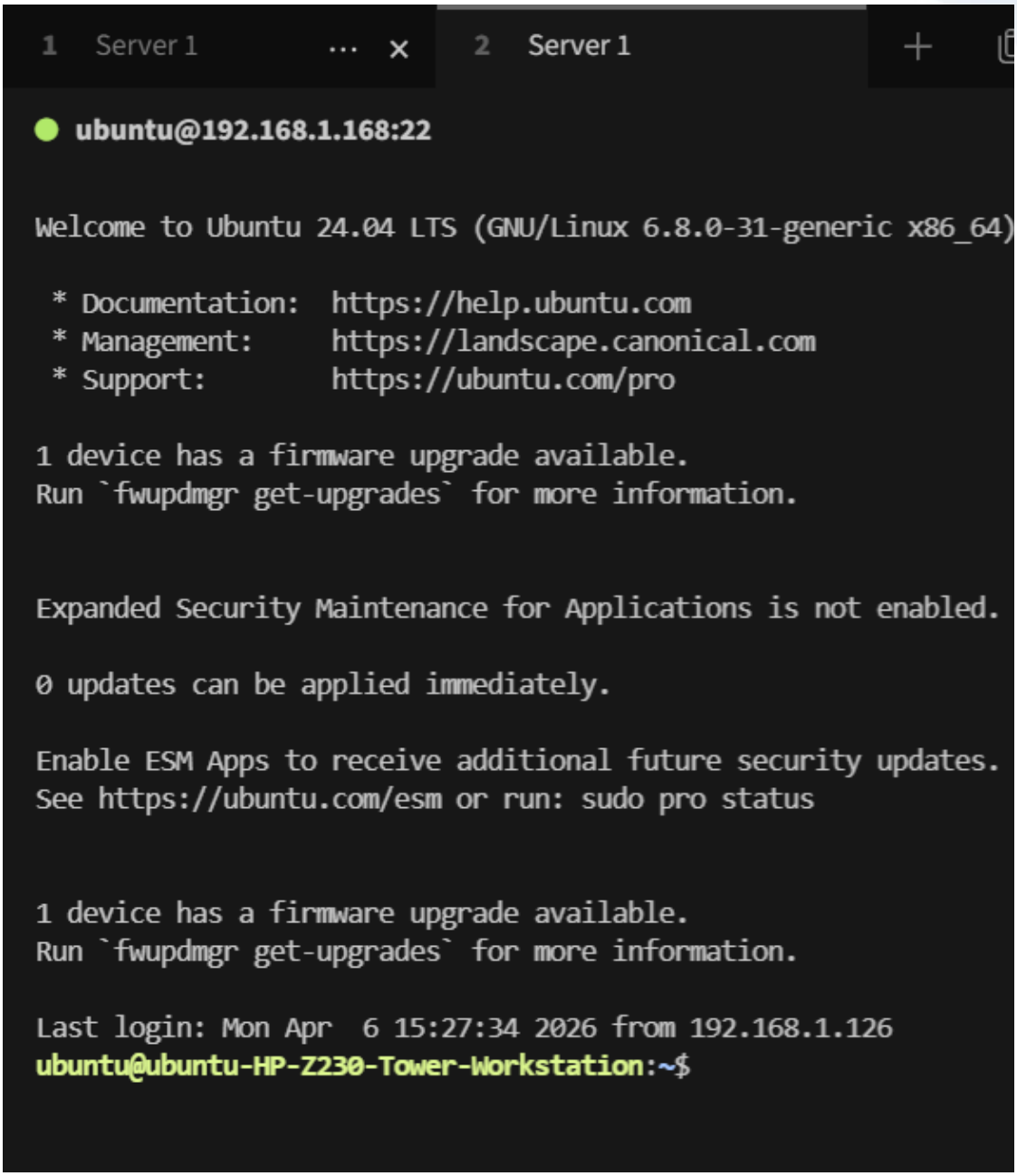
Mục tiêu của bước này là tăng cường bảo mật cho hệ thống bằng cách vô hiệu hóa đăng nhập bằng mật khẩu trong OpenSSH, chỉ cho phép truy cập thông qua SSH key. Việc này giúp hạn chế nguy cơ bị tấn công dò mật khẩu và nâng cao mức độ an toàn cho server, đặc biệt khi hệ thống được triển khai trên môi trường mạng công khai.

**Hướng dẫn thực thi:**

***Bước 1: Copy public key vào thư mục .ssh trên server bằng FileZilla như đã hướng dẫn.***




***Bước 2: Kết nối vào server bằng Tabby terminal bằng private key (đã hướng dẫn).***




***Bước 3: Cài vim ( công cụ hỗ trợ chỉnh sửa file) bằng lệnh***

```
sudo apt update

sudo apt install vim -y
```

***Bước 4: Mở file sshd_config để cấu hình ssh bằng lệnh***

```
sudo vim /etc/ssh/sshd_config
```

Sau khi có thành công, bạn sẽ xem được nội dung trong sshd_config

```
1 Settings        2 Server 1
ubuntu@192.168.1.168:22                    Reconnect

#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
#GSSAPIStrictAcceptorCheck yes
#GSSAPIKeyExchange no

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the KbdInteractiveAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via KbdInteractiveAuthentication may bypass
# the setting of "PermitRootLogin prohibit-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and KbdInteractiveAuthentication to 'no'.
UsePAM yes

#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
PrintMotd no
#PrintLastLog yes
#TCPKeepAlive yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none
```

***Bước 5: Chỉnh sửa các dòng sau***

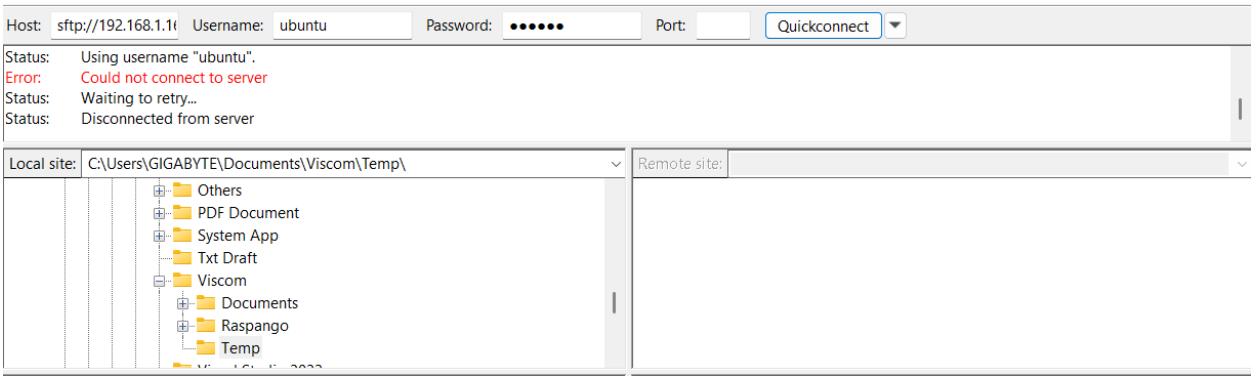
```
PasswordAuthentication no

PermitRootLogin no
```

Sau khi chỉnh các dòng trên về no, bạn nhấn nút “Esc” trên bàn phím. Sau đó nhấn “:wq” và “Enter” để lưu thay đổi.

***Bước 6: Khởi động lại SSH bằng lệnh***

```
sudo systemctl restart ssh
```

***Bước 7: Kiểm tra kết nối bằng password bằng FileZilla, bạn sẽ thấy không kết nối được.***




# Bật firewall

Trong Ubuntu, tường lửa được quản lý bằng UFW (Uncomplicated Firewall). Việc bật firewall giúp kiểm soát các kết nối ra vào hệ thống, chỉ cho phép những dịch vụ được cấu hình trước hoạt động. Trước khi kích hoạt, cần cho phép các dịch vụ cần thiết như SSH để tránh mất kết nối từ xa. Điều này giúp tăng cường bảo mật và hạn chế các truy cập trái phép vào server.

**Hướng dẫn cài đặt:**

***Bước 1: Cài đặt UFW bằng lệnh***

```
sudo apt insall ufw -y
```

***Bước 2: Cho phép SSH vào server bằng lệnh***

```
sudo ufw allow OpenSSH
```

***Bước 3: Bật Firewall bằng lệnh***

```
sudo ufw enable
```

***Bước 4: Kiểm tra lại trạng thái Firewall***

```
sudo ufw status
```

Sau khi cài đặt và cấu hình thành công, bạn sẽ thấy :

```
ubuntu@ubuntu-HP-Z230-Tower-Workstation:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)

ubuntu@ubuntu-HP-Z230-Tower-Workstation:~$
```

## Cài đặt git

Git là hệ thống quản lý phiên bản giúp theo dõi thay đổi mã nguồn, hỗ trợ làm việc nhóm và quản lý các phiên bản của dự án. Git cho phép lưu lịch sử chỉnh sửa, quay lại phiên bản trước và làm việc với nhiều nhánh (branch), đồng thời dễ dàng đồng bộ với các nền tảng như GitHub hoặc GitLab.

**Hướng dẫn cài đặt:**

***Bước 1: Cài đặt git bằng lệnh sau***

```
sudo apt update

sudo apt install git -y
```

***Bước 2: Kiểm ta lại phiên bản cài đặt***

```
git --version
```

Sau khi cài đặt xong git, bạn sẽ thấy màn hình sau

```
ubuntu@ubuntu-HP-Z230-Tower-Workstation:~$ git --version
git version 2.43.0
ubuntu@ubuntu-HP-Z230-Tower-Workstation:~$
```

## Cài đặt nginx

Nginx là một web server dùng để xử lý các yêu cầu từ trình duyệt (HTTP/HTTPS) và trả về nội dung web. Ngoài ra, Nginx còn có thể chuyển tiếp yêu cầu đến các ứng dụng phía sau (reverse proxy), giúp hệ thống hoạt động ổn định và hiệu quả hơn khi triển khai thực tế.

***Hướng dẫn cài đặt:***

***Bước 1: Cài đặt Nginx bằng lệnh sau:***

```
sudo apt update

sudo apt install nginx -y
```

***Bước 2: Chạy Nginx ngay lập tức bằng lệnh***

```
sudo systemctl start nginx
```

***Bước 3: Đảm bảo nginx luôn tự động chạy mỗi khi server khởi động bằng lệnh***

```
sudo systemctl enable nginx
```

***Bước 4: Kiểm tra lại trạng thái của nginx bằng lệnh***

```
sudo systemctl status nginx
```

Khi tải thành công nginx, bạn sẽ thấy

```
ubuntu@ubuntu-HP-Z230-Tower-Workstation:~$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Mon 2026-04-06 16:50:20 +07; 37s ago
       Docs: man:nginx(8)
    Process: 6154 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code>
    Process: 6156 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, >
   Main PID: 6157 (nginx)
      Tasks: 5 (limit: 18812)
     Memory: 3.7M (peak: 4.3M)
        CPU: 16ms
     CGroup: /system.slice/nginx.service
             ├─6157 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─6158 "nginx: worker process"
             ├─6159 "nginx: worker process"
             ├─6160 "nginx: worker process"
             └─6161 "nginx: worker process"

Apr 06 16:50:20 ubuntu-HP-Z230-Tower-Workstation systemd[1]: Starting nginx.service - A hig>
Apr 06 16:50:20 ubuntu-HP-Z230-Tower-Workstation systemd[1]: Started nginx.service - A high>
lines 1-19/19 (END)
```

# Chương 5: Deploy Raspango

Chương này trình bày các bước triển khai hệ thống **Raspango** lên server một cách tuần tự. Quy trình bao gồm chuẩn bị thư mục, lấy mã nguồn, cấu hình uWSGI và Nginx, thiết lập môi trường, cài đặt các thư viện cần thiết và khởi động web server. Người đọc có thể thực hiện lần lượt theo các bước để triển khai hệ thống một cách nhanh chóng và hiệu quả.

## Nội dung


26. Tạo folder cần thiết
27. Clone source code
28. Cài đặt Environment
29. Cài đặt package
30. Copy config uwsgi và nginx
31. Start webserver

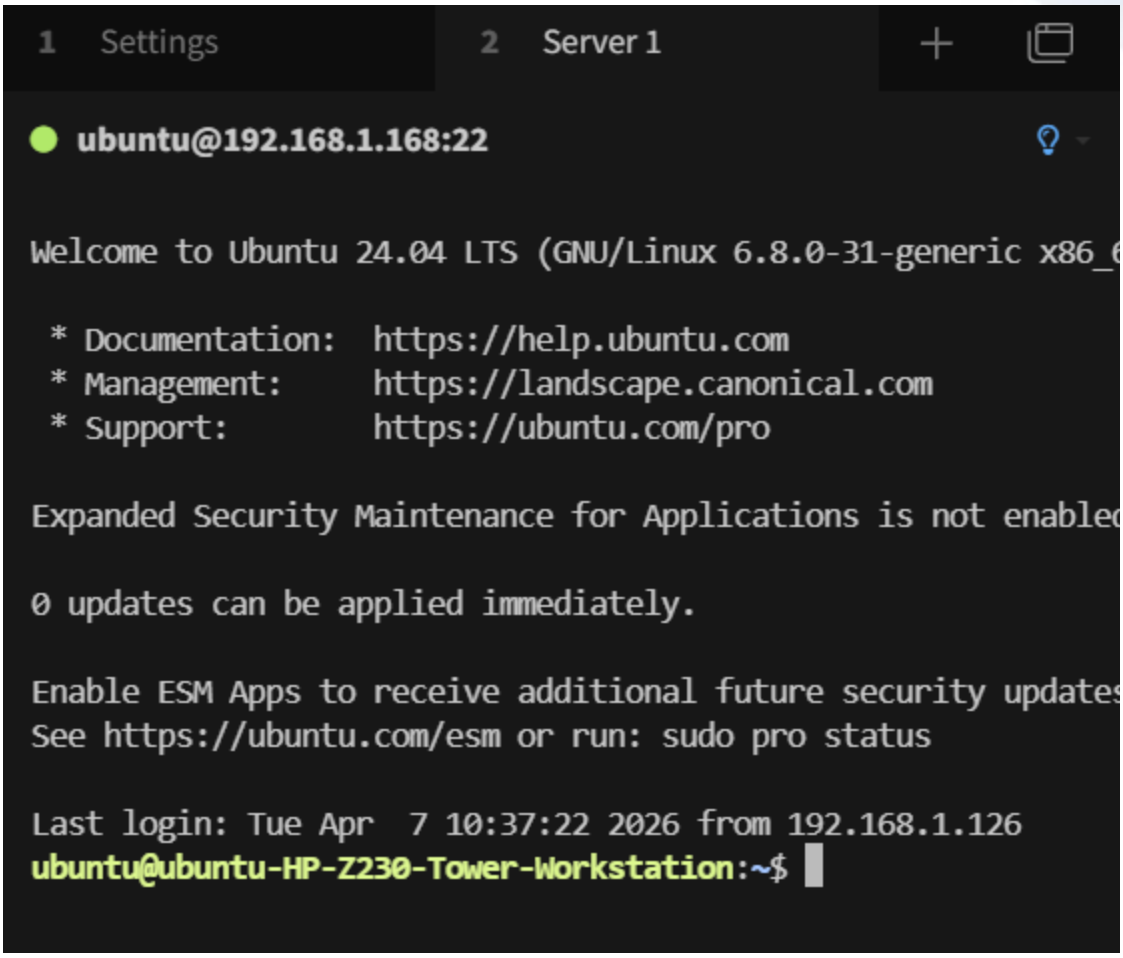

---

## Tạo folder cần thiết

Trước khi triển khai hệ thống, cần tạo cấu trúc thư mục phù hợp trên server để lưu trữ mã nguồn và các thành phần liên quan. Việc tổ chức thư mục đúng chuẩn giúp dễ quản lý, đảm bảo quyền truy cập và thuận tiện cho việc cấu hình các dịch vụ như Nginx và uWSGI. Trong bước này, thư mục chính của project sẽ được tạo trong hệ thống và thiết lập quyền cần thiết để phục vụ cho các bước triển khai tiếp theo.

**Hướng dẫn thực hiện:**

***Bước 1: Dùng Tabby terminal để truy cập vào server***




***Bước 2: Quay về thư mục gốc bằng lệnh***

```
cd /
```

***Bước 3: Tạo thư mục project bằng lệnh***

```
sudo mkdir -p  /var/www/raspango
```

*Lưu ý: /var/www/ thường là thư mục dùng để lưu project*

***Bước 4: Cấp quyền cho user của bạn bằng lệnh***

```
sudo chown -R  $USER:$USER  /var/www
```

*Lưu ý: Mặc định, nhiều lệnh hệ thống yêu cầu phải thêm sudo (ví dụ: sudo apt install, sudo systemctl restart nginx), vì chúng cần quyền quản trị (root). Lệnh này dùng để chuyển toàn bộ thư mục /var/www/ về quyền sở hữu của user hiện tại, giúp bạn chỉnh sửa file mà không cần dùng sudo.*

***Bước 5: Đi vào thư mục  bằng lệnh***

```
cd /var/www/raspango
```

***Bước 6: Kiểm tra thư mục bằng lệnh***

```
pwd
```

Nếu tạo thành công, bạn sẽ thấy

```
ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ pwd
/var/www/raspango
```

## Clone source code

Sau khi đã chuẩn bị thư mục làm việc, bước tiếp theo là tải mã nguồn của dự án từ hệ thống quản lý phiên bản về server. Việc clone source code giúp đảm bảo toàn bộ cấu trúc và lịch sử dự án được đồng bộ chính xác. Người dùng có thể sử dụng giao thức HTTPS hoặc SSH tùy theo quyền truy cập, sau đó kiểm tra lại các tệp tin để xác nhận mã nguồn đã được tải thành công.

**Source code:**

```
https://github.com/thigiacmaytinh/Raspango.git
```

**Hướng dẫn thực hiện**

***Bước 1: Clone source code trên về máy bằng lệnh sau***

```
git clone https://github.com/thigiacmaytinh/Raspango.git .
```

*Lưu ý: Dấu "." cuối để clone trực tiếp vào thư mục hiện tại, không tạo folder con.*

***Bước 2: Kiểm tra lại bằng lệnh "ls". Nếu clone thành công, bạn sẽ thấy:***

```
ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ ls
install_python_library.bat  README.md  requirements.txt  server  start.bat  start.sh
```

## Cài đặt Environtment

Việc cài đặt môi trường giúp thiết lập các thư viện và công cụ cần thiết cho dự án trong một không gian riêng biệt, tránh xung đột với hệ thống. Nhờ đó, việc phát triển và triển khai ứng dụng trở nên ổn định, dễ quản lý và nhất quán hơn giữa các môi trường khác nhau.

**Hướng dẫn thực hiện**

***Bước 1: Kiểm tra môi trường python bằng lệnh***

```
python3 --version
```

Bạn sẽ thấy phiên bản python được cài mặc định cho ubuntu

```
ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ python3 --version
Python 3.12.3
```

***Bươc 2: Cài đặt pip để tải các gói thư viện***

```
sudo apt install python3-venv python3-pip -y
```

***Bước 3: Tạo môi trường ảo bằng lệnh***

```
python3 -m venv venv
```

***Bước 4: Kích hoạt môi trường ảo vừa tạo bằng lệnh***

```
source venv/bin/activate
```

Sau khi kích hoạt, bạn sẽ thấy

```
ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ python3 -m venv venv
ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ source venv/bin/activate
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ s
```

***Bước 5: Cài đặt uWSGI bằng lệnh***

```
pip install uwsgi
```

Sau khi cài đặt thành công, bạn sẽ thấy

```
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ pip install uwsgi
Collecting uwsgi
  Downloading uwsgi-2.0.31.tar.gz (822 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 822.8/822.8 kB 4.5 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: uwsgi
  Building wheel for uwsgi (pyproject.toml) ... done
  Created wheel for uwsgi: filename=uwsgi-2.0.31-cp312-cp312-linux_x86_64.whl size=52340
4 sha256=5fc7eb45f7b6e70191f514ee2d3060abdce1752a52bad522b34ff972d600de4c
  Stored in directory: /home/ubuntu/.cache/pip/wheels/de/9b/b9/cc2e42ac5a3197a4be493c415
bcc29ea7e58fbfa4b715946a5
Successfully built uwsgi
Installing collected packages: uwsgi
Successfully installed uwsgi-2.0.31
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$
```

# Cài đặt package

Việc cài đặt package là một bước quan trọng trong quá trình thiết lập môi trường phát triển, giúp bổ sung các thư viện và công cụ cần thiết cho dự án. Các package cung cấp những chức năng sẵn có, giúp lập trình viên không cần phải xây dựng lại từ đầu, từ đó tiết kiệm thời gian và giảm độ phức tạp khi phát triển. Đồng thời, việc quản lý package đúng cách còn giúp đảm bảo tính nhất quán giữa các môi trường, hạn chế lỗi phát sinh do thiếu hoặc sai phiên bản thư viện, đặc biệt trong các dự án có nhiều phụ thuộc.

**Hướng dẫn thực hiện**

***Bước 1: Cài đặt các package đi kèm. Đầu tiên vào file requirements.txt bằng lệnh "vim requirements.txt"và sửa lại nội dung như sau***

```
Django==2.2.4

djangorestframework

django-cors-headers

django-crontab==0.7.1

python-dateutil==2.9.0.post0

PyJWT==2.12.1

nose==1.3.7

Sphinx==7.2.6

sphinx-rtd-theme==2.0.0

openpyxl==3.1.2

Pillow==10.4.0

numpy==1.26.4

Click==8.1.7

protobuf==3.20.3

urllib3==1.26.18

opencv-python==4.8.1.78

tensorflow==2.16.1

ultralytics
```

Sau đó nhấn “Esc”, gõ “:wq” và Enter.

***Bước 2: Để tải các package, bạn dùng lệnh***

```
pip install --upgrade pip

pip install -r requirements.txt
```

Nếu cài đặt thành công, bạn sẽ thấy

```
d7ecf9cee688
Successfully built django-crontab
Installing collected packages: nvidia-cusparselt-cu13, nose, namex, mpmath, libclang, f
da-toolkit, wrapt, urllib3, typing-extensions, triton, termcolor, tensorboard-data-serv
parse, sphinxcontrib-serializinghtml, sphinxcontrib-qthelp, sphinxcontrib-jsmath, sphin
elp, sphinxcontrib-devhelp, sphinxcontrib-applehelp, snowballstemmer, six, setuptools,
ing, PyJWT, Pygments, psutil, protobuf, polars-runtime-32, Pillow, packaging, opt-einsu
, nvidia-nvshmem-cu13, nvidia-nvjitlink, nvidia-nccl-cu13, nvidia-curand, nvidia-cufile
runtime, nvidia-cuda-nvrtc, nvidia-cuda-cupti, nvidia-cublas, numpy, networkx, mdurl, M
kdown, kiwisolver, imagesize, idna, gast, fsspec, fonttools, filelock, et-xmlfile, docu
cuda-pathfinder, cmake, Click, charset_normalizer, certifi, babel, asgiref, alabaster,
, werkzeug, scipy, requests, python-dateutil, polars, optree, openpyxl, opencv-python,
e, nvidia-cufft, nvidia-cudnn-cu13, ml-dtypes, markdown-it-py, Jinja2, h5py, grpcio, go
ango, cuda-bindings, contourpy, tensorboard, Sphinx, rich, nvidia-cusolver, matplotlib,
mework, django-crontab, django-cors-headers, astunparse, sphinxcontrib-jquery, keras, t
ow, sphinx-rtd-theme, ultralytics-thop, torchvision, ultralytics
Successfully installed Click-8.1.7 Django-4.2.11 Jinja2-3.1.6 MarkupSafe-3.0.3 Pillow-1
12.1 Pygments-2.20.0 Sphinx-7.2.6 absl-py-2.4.0 alabaster-0.7.16 asgiref-3.11.1 astunpa
l-2.18.0 certifi-2026.2.25 charset_normalizer-3.4.7 cmake-4.3.1 contourpy-1.3.3 cuda-bi
cuda-pathfinder-1.5.2 cuda-toolkit-13.0.2 cycler-0.12.1 django-cors-headers-4.9.0 djang
1 djangorestframework-3.17.1 docutils-0.20.1 et-xmlfile-2.0.0 filelock-3.25.2 flatbuffe
nttools-4.62.1 fsspec-2026.3.0 gast-0.7.0 google-pasta-0.2.0 grpcio-1.80.0 h5py-3.16.0
esize-2.0.0 keras-3.14.0 kiwisolver-1.5.0 libclang-18.1.1 markdown-3.10.2 markdown-it-p
tlib-3.10.8 mdurl-0.1.2 ml-dtypes-0.4.1 mpmath-1.3.0 namex-0.1.0 networkx-3.6.1 nose-1.
.4 nvidia-cublas-13.1.0.3 nvidia-cuda-cupti-13.0.85 nvidia-cuda-nvrtc-13.0.88 nvidia-cu
0.96 nvidia-cudnn-cu13-9.19.0.56 nvidia-cufft-12.0.0.61 nvidia-cufile-1.15.1.6 nvidia-c
5 nvidia-cusolver-12.0.4.66 nvidia-cusparse-12.6.3.3 nvidia-cusparselt-cu13-0.8.0 nvidi
28.9 nvidia-nvjitlink-13.0.88 nvidia-nvshmem-cu13-3.4.5 nvidia-nvtx-13.0.85 opencv-pyth
penpyxl-3.1.2 opt-einsum-3.4.0 optree-0.19.0 packaging-26.0 polars-1.39.3 polars-runtim
otobuf-4.25.3 psutil-7.2.2 pyparsing-3.3.2 python-dateutil-2.9.0.post0 pyyaml-6.0.3 req
ich-14.3.3 scipy-1.17.1 setuptools-81.0.0 six-1.17.0 snowballstemmer-3.0.1 sphinx-rtd-t
inxcontrib-applehelp-2.0.0 sphinxcontrib-devhelp-2.0.0 sphinxcontrib-htmlhelp-2.1.0 sph
ery-4.1 sphinxcontrib-jsmath-1.0.1 sphinxcontrib-qthelp-2.0.0 sphinxcontrib-serializing
parse-0.5.5 sympy-1.14.0 tensorboard-2.18.0 tensorboard-data-server-0.7.2 tensorflow-2.
-3.3.0 torch-2.11.0 torchvision-0.26.0 triton-3.6.0 typing-extensions-4.15.0 ultralytic
lytics-thop-2.0.18 urllib3-2.2.2 werkzeug-3.1.8 wheel-0.46.3 wrapt-2.1.2
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ vim requirements.txt
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$
```

## Copy config uwsgi và nginx

Sau khi đã cài đặt các package, bước tiếp theo là cấu hình uWSGI và Nginx để triển khai ứng dụng lên môi trường server. uWSGI đóng vai trò là tầng trung gian dùng để chạy ứng dụng Python và xử lý logic backend, trong khi Nginx hoạt động như một web server phía trước, tiếp nhận request từ người dùng và chuyển tiếp đến uWSGI thông qua socket. Các tệp cấu hình cần được thiết lập để xác định đường dẫn ứng dụng, cơ chế giao tiếp và cách phân phối request, từ đó đảm bảo hệ thống hoạt động ổn định, hiệu quả và có khả năng mở rộng.

**Hướng dẫn thực hiện**

***Bước 1: Tạo file raspango.ini trong thư mục raspango bằng vim***

```
vim raspango.ini
```

Sau đó nhấn I để vào chế độ insert và dán nội dung sau vào

```
[uwsgi]

chdir = /var/www/raspango/server

module = raspango.wsgi:application

master = true

processes = 5

socket = /var/www/raspango/raspango.sock

chmod-socket = 666

vacuum = true

die-on-term = true
```

Sau đó nhấn “ESC”, gõ “:wq” và enter

***Bước 2: Tạo config Nginx bằng vim***

```
sudo vim /etc/nginx/sites-available/raspango
```

Nhấn i và dán nội sung sau

```
server                                                                  {
    listen                                                             80;
    server_name                                            your_ip_or_domain;

    location                                /                           {
      include                                                uwsgi_params;
      uwsgi_pass                     unix:/var/www/raspango/raspango.sock;
    }
}
```

Sau đó nhấn "Esc", gõ ":wq" và enter

***Bước 3: Enable config nginx***

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

sudo nginx -t

sudo systemctl restart nginx
```

*Lưu ý:*

- *Lệnh đầu tiên tạo một symbolic link từ config gốc sang thư mục sites-enabled để nginx đọc config vừa tạo ( symbolic là một loại shortcut trong Linux, nó dùng để trỏ đến file hoặc thư mục gốc)*
- *Lệnh thứ 2 là kiểm tra nginx*
- *Lệnh thứ 3 là khởi động nginx lại*

Toàn bộ quá trình config nginx sẽ như sau

```
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/etc/nginx/sites-available$ sudo v
/nginx/sites-available/raspango
[sudo] password for ubuntu:
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/etc/nginx/sites-available$ sudo l
tc/nginx/sites-available/raspango /etc/nginx/sites-enabled/
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/etc/nginx/sites-available$ sudo n
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/etc/nginx/sites-available$ sudo s
l restart nginx
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/etc/nginx/sites-available$
```

***Bước 4: Chạy uWSGI***

- ***Cách 1: Chạy thủ công bằng lệnh***

```
uwsgi --ini raspango.ini --daemonize uwsgi.log
```

- ***Cách 2: Chạy uWSGI như service (mỗi lần server khởi động là uwsgi tự chạy)***

Tạo file service:

```
sudo vim /etc/systemd/system/uwsgi.service
```

Nhấn "I" và dán nội dung

```
[Unit]
Description=uWSGI                                                    service
After=network.target

[Service]
User=ubuntu
WorkingDirectory=/var/www/raspango
ExecStart=/var/www/raspango/venv/bin/uwsgi        --ini        raspango.ini
Restart=always

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

Sau đó nhấn "ESC", gõ ":wq" và Enter

Enable service (đảm bảo mỗi lần server khởi động là uwsgi chạy)

```
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable uwsgi
sudo systemctl start uwsgi
```

Giải thích

- **sudo systemctl daemon-reexec: Khởi** động lại tiến trình systemd (hệ điều hành service manager) mà không reboot máy
- **sudo systemctl daemon-reload:** Yêu cầu systemd đọc lại toàn bộ file cấu hình service
- **sudo systemctl enable uwsgi:** Cấu hình cho service tự động chạy khi server khởi động (boot)
- **sudo systemctl start uwsgi:** Khởi động service ngay lập tức

Kiểm tra trạng thái bằng lệnh

```
sudo systemctl status uwsgi
```

Trạng thái chuyển sang active có nghĩa là đã thành công

```
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/etc/systemd/system$ sudo systemc
wsgi
● uwsgi.service - uWSGI service
     Loaded: loaded (/etc/systemd/system/uwsgi.service; enabled; preset: enabled
     Active: active (running) since Wed 2026-04-08 15:58:38 +07; 9s ago
   Main PID: 5734 (uwsgi)
      Tasks: 6 (limit: 18693)
     Memory: 47.0M (peak: 47.3M)
        CPU: 508ms
     CGroup: /system.slice/uwsgi.service
             ├─5734 /var/www/raspango/venv/bin/uwsgi --ini raspango.ini
             ├─5735 /var/www/raspango/venv/bin/uwsgi --ini raspango.ini
             ├─5736 /var/www/raspango/venv/bin/uwsgi --ini raspango.ini
             ├─5737 /var/www/raspango/venv/bin/uwsgi --ini raspango.ini
             ├─5738 /var/www/raspango/venv/bin/uwsgi --ini raspango.ini
             └─5739 /var/www/raspango/venv/bin/uwsgi --ini raspango.ini

Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: mapped 437424 byte
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: *** Operational MO
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: WSGI app 0 (mountp
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: *** uWSGI is runni
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: spawned uWSGI mast
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: spawned uWSGI work
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: spawned uWSGI work
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: spawned uWSGI work
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: spawned uWSGI work
Apr 08 15:58:38 ubuntu-HP-Z230-Tower-Workstation uwsgi[5734]: spawned uWSGI work
lines 1-25/25 (END)
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/etc/systemd/system$
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/etc/systemd/system$ ps aux | gre
ubuntu      5734  1.6  0.3  73792 60268 ?        Ss   15:58   0:00 /var/www/rasp
```

***Bước 5: Mở port 80 trên Firewall bằng lệnh***

```
sudo ufw allow 80/tcp
```

***Bước 6: Reload lại Firewall bằng lệnh***

```
sudo ufw reload
```

Sau khi mở thành công dùng lệnh

```
sudo ufw status
```

Sau khi mở thành công, bạn sẽ thấy

```
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ sudo uf
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
```

## Start webserver

Để website hoạt động, bạn cần "khởi chạy" cả ứng dụng backend và web server. Trước hết, uWSGI sẽ đóng vai trò chạy ứng dụng của bạn (ví dụ Django/Flask) và xử lý các request từ Nginx. Vì ở trên đã hướng dẫn bạn cấu hình để uWSGI trở thành service của hệ thống (server khởi động thì service này sẽ chạy) nên để khởi động lại uWSGI bạn chỉ cần nhập lệnh

```
sudo systemctl restart uwsgi
```

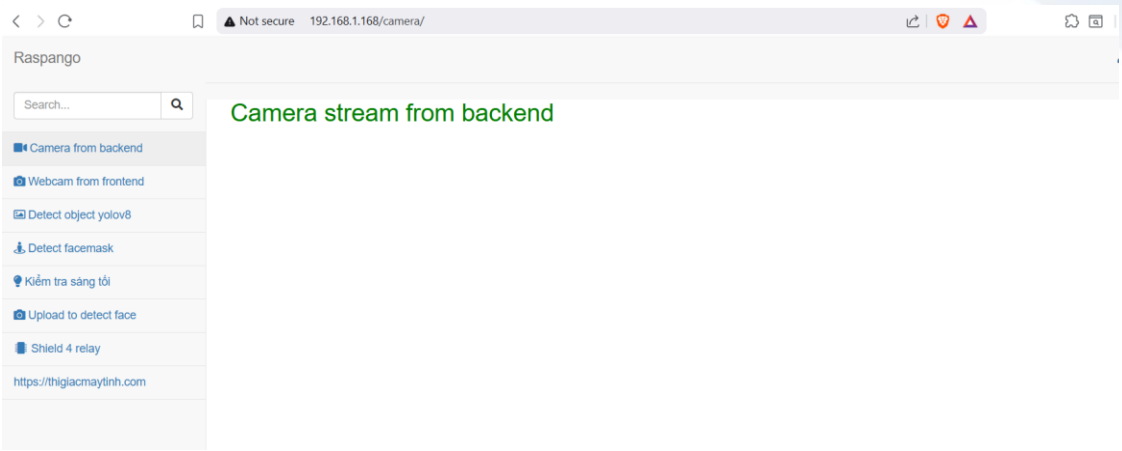
Sau đó, bạn cần khởi động lại Nginx bằng lệnh sudo systemctl restart nginx. Nginx sẽ đóng vai trò nhận request từ người dùng (qua trình duyệt), rồi chuyển tiếp đến uWSGI để xử lý và trả kết quả về lại cho người dùng. Khi cả hai thành phần đều chạy đúng, website của bạn sẽ có thể truy cập bình thường qua trình duyệt.

```
sudo systemctl restart nginx
```

Mở trình duyệt và kiểm tra với địa chỉ IP của server

```
http://<IP>:80
```

Khi thành công, giao diện trình duyệt của bạn sẽ như sau


Not secure 192.168.1.168/camera/
Raspango
Search...
Camera from backend
Webcam from frontend
Detect object yolov8
Detect facemask
Kiểm tra sáng tối
Upload to detect face
Shield 4 relay
https://thigiacmaytinh.com
Camera stream from backend

# Hướng dẫn cấu hình web server

Chương này trình bày cách Web Server xử lý một request từ người dùng, thông qua sự phối hợp giữa Nginx (tiếp nhận và điều phối request) và uWSGI (kết nối đến backend Python). Đồng thời, chương cũng hướng dẫn cách cấu hình các thành phần này để triển khai ứng dụng web một cách đơn giản và hiệu quả.

## Nội dung


32. Cách web server handle 1 request
33. Hướng dẫn config web server


---

## Cách web server handle 1 request

Để hiểu cách một website hoạt động, trước hết cần nắm được cách Web Server xử lý một request từ người dùng. Khi client gửi yêu cầu, Nginx sẽ tiếp nhận và quyết định cách xử lý: trả về dữ liệu trực tiếp hoặc chuyển tiếp sang uWSGI để gọi đến ứng dụng backend. Phần này sẽ trình bày chi tiết luồng xử lý đó, giúp làm rõ cách các thành phần trong hệ thống phối hợp với nhau để tạo ra phản hồi cuối cùng.

> *Nginx đọc là "en gin X"*

**Tổng quan vai trò các thành phần:**

- **Client (trình duyệt):** gửi request từ người dùng
- **Nginx:** tiếp nhận request, xử lý file tĩnh và điều phối
- **uWSGI:** trung gian chuyển request vào ứng dụng Python
- **Backend:** Xử lý logic và tạo response

**Luồng xử lý chi tiết:**

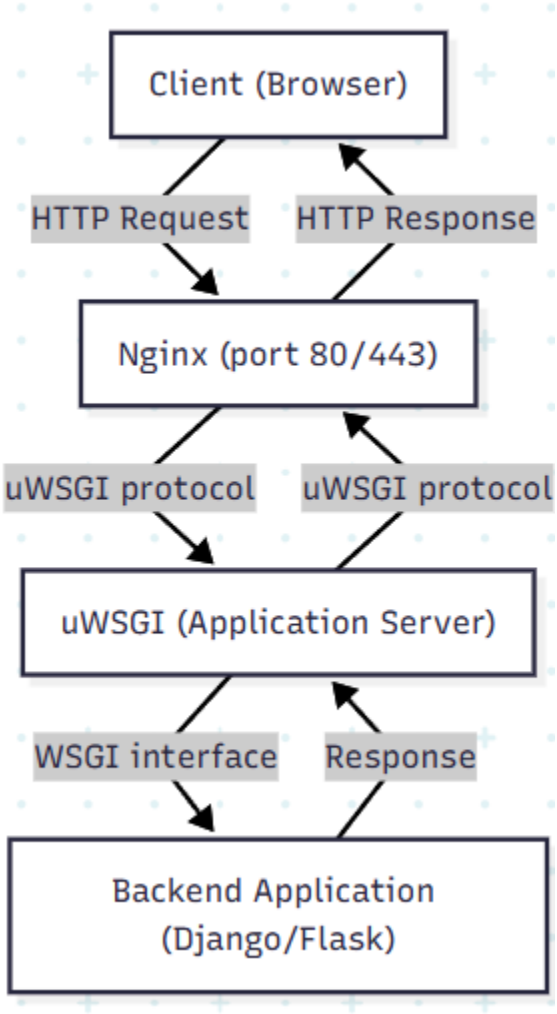
1. Client gửi HTTP Request

Khi người dùng truy cập một URL như : http://your-domain.com/user .Trình duyệt sẽ tạo một HTTP request bao gồm: Method, URL, Header, Body đến server.

2. Nginx tiếp nhận request tại server
   Nginx lắng nghe tại port 80 (HTTP) hoặc 443 (HTTPS), nhận request từ client và dựa vào cấu hình (server_name, location) để xác định cách xử lý phù hợp.
3. Nginx phân loại request
   - ***Static request (file tĩnh):*** Nginx sẽ truy xuất trực tiếp file từ hệ thống lưu trữ và trả về cho client mà không cần thông qua backend. Cách xử lý này giúp tốc độ phản hồi nhanh hơn và giảm tải đáng kể cho server ứng dụng. *Ví dụ: /static/logo.png, /css/style.css*
   - ***Dynamic request (yêu cầu động):*** Đây là các request cần xử lý logic, vì vậy Nginx sẽ chuyển tiếp đến backend thông qua uWSGI. Backend sẽ tiếp nhận, xử lý dữ liệu và tạo response trước khi gửi ngược lại cho người dùng. *Ví dụ: /user, /api/data*
4. Nginx chuyển request qua uWSGI
   Thông qua cấu hình "uwsgi_pass unix:/var/www/raspango/raspango.sock;" (đã cấu hình ở chương trước). Request được gửi qua socket (một kênh giao tiếp nội bộ trong hệ thống). Cách xử lý này nhanh hơn so với việc gửi qua HTTP thông thường.
5. uWSGI kết nối với ứng dụng Python
   uWSGI nhận request, chuyển đổi sang định dạng mà Python hiểu được và gọi vào entry point "raspango.wsgi:application"
6. Backend xử lý nghiệp vụ
   Ứng dụng backend sẽ:
   - Phân tích request
   - Xử lý logic (authentication, business logic, ...)
   - Tương tác database (nếu cần)
   - Tạo response (HTML, JSON, ...)
7. Response trả ngược về client
   *Luồng trả về: Backend → uWSGI → Nginx → Browser*
   - uWSGI gửi response lại cho Nginx

- Nginx đóng gói thành HTTP response
- Trả về trình duyệt để hiển thị

**Tóm tắt luồng xử lý:**




## Hướng dẫn config web server

Để triển khai một ứng dụng web lên server, cần cấu hình để các thành phần trong hệ thống có thể phối hợp với nhau. Trong mô hình này, Nginx đóng vai trò tiếp nhận request từ người dùng và điều phối, trong khi uWSGI đảm nhiệm việc chạy ứng dụng Python và xử lý logic backend. Phần này sẽ hướng dẫn cách thiết

lập hai thành phần trên để chúng "kết nối" với nhau thông qua socket, giúp hệ thống hoạt động ổn định và hiệu quả.

**Các bước thực hiện:**

***Bước 1: Tạo file cấu hình uWSGI bằng lệnh***

```
vim /var/www/raspango/raspango.ini
```

**Bước 2: Thêm nội dung vào file**

```
[uwsgi]

chdir = /var/www/raspango/code/server

module = raspango.wsgi:application

master = true

processes = 5



socket = /var/www/raspango/raspango.sock

chmod-socket = 666



vacuum = true

die-on-term = true
```

**Giải thích ý nghĩa:**

- ***chdir = /var/www/raspango/server*:** Thiết lập thư mục làm việc của ứng dụng. uWSGI sẽ chạy trong thư mục này để tìm source code

- ***module = raspango.wsgi:application*:** Xác định điểm bắt đầu của ứng dụng Django. Đây là nơi uWSGI gọi vào để chạy backend.
- ***master = true:*** Bật tiến trình master. Quản lý các worker process, giúp hệ thống ổn định hơn
- ***processes = 5:*** Số lượng tiến trình xử lý request. Cho phép xử lý nhiều request đồng thời.
- ***socket = /var/www/raspango/raspango.sock:*** Tạo socket để giao tiếp với Nginx. Đây là "đường ống" truyền request giữa Nginx và uWSGI
- ***chmod-socket = 666:*** Cấp quyền cho socket. Đảm bảo Nginx có thể đọc/ghi để giao tiếp với uWSGI
- ***vacuum = true:*** Tự động xóa file socket khi dừng uWSGI. Tránh lỗi do socket cũ còn tồn tại
- ***die-on-term = true:*** Cho phép uWSGI tắt đúng cách khi hệ thống gửi tín hiệu dừng. Giúp quản lý service tốt hơn (systemd, docker, ...)

**File cấu hình này giúp:**

- Xác định vị trí ứng dụng
- Khởi động backend Django
- Cho phép xử lý nhiều request
- Thiết lập giao tiếp với Nginx qua socket

**Bước 3: Tạo file cấu hình Nginx bằng lệnh**

```
sudo vim /etc/nginx/sites-available/raspango
```

**Bước 4: Thêm nội dung vào file vừa tạo**

```
server                                                                  {
    listen                                                             80;
    server_name                                             your_ip_or_domain;
```

```
    location / {

      alias /var/www/raspango/code/server/web/static/;

    }

    location / {

      alias /var/www/raspango/media/;

    }

    location                                /                                {
      include                                                     uwsgi_params;
      uwsgi_pass                      unix:/var/www/raspango/raspango.sock;
    }
}
```

**Ý nghĩa từng thành phần**

- ***server { ... } :*** Khai báo một server block. Mỗi block tương ứng với một website mà Nginx quản lý
- **listen 80; :** Nginx lắng nghe tại port 80 (HTTP). Khi người dùng truy cập web (không HTTPS), request sẽ vào đây.
- **server_name your_ip_or_domain; :** Xác định domain hoặc IP của server. Nginx dùng để phân biệt các website khác nhau
- **location / { ... } :** Xử lý tất cả request bắt đầu từ /. Nghĩa là mọi request gửi đến website đều đi qua đây
- **include uwsgi_params; :** Nạp các tham số chuẩn để giao tiếp với uWSGI. Bao gồm thông tin như method, header, path,...
- **uwsgi_pass unix:/var/www/raspango/raspango.sock; :** Chuyển request đến uWSGI thông qua socket

**Bước 5: Kích hoạt cấu hình**

```
sudo ln -s /etc/nginx/sites-available/raspango /etc/nginx/sites-enabled/
sudo                                nginx                                -t
sudo systemctl restart nginx
```

# Chương 7 – Các lỗi thường gặp

Trong quá trình triển khai web, một số lỗi như server không truy cập được, 503 hay 404 có thể xảy ra do cấu hình hoặc dịch vụ chưa đúng. Chương này trình bày các lỗi phổ biến khi sử dụng Nginx và uWSGI, đồng thời hướng dẫn cách xem log để nhanh chóng tìm và khắc phục nguyên nhân.

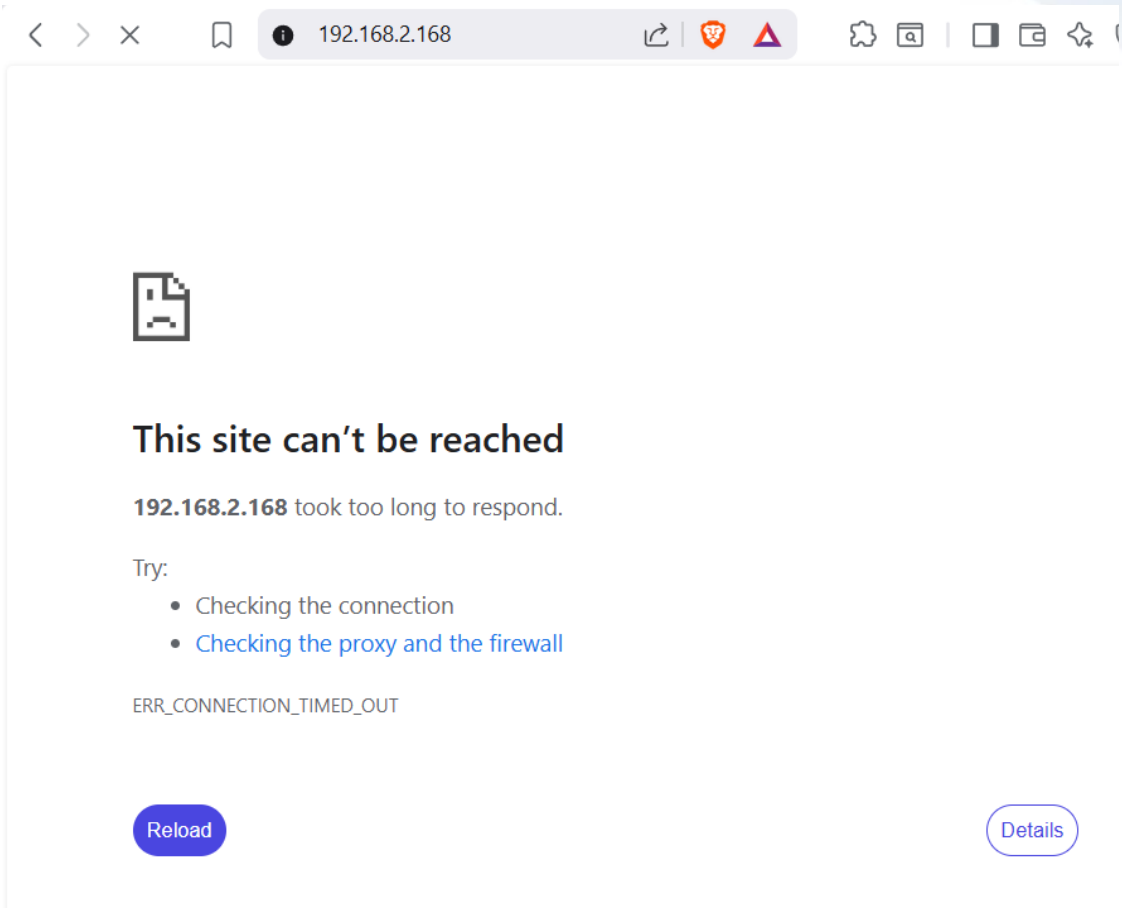
## Nội dung


34. Server unreachable
35. Lỗi 503
36. Lỗi 404
37. Cách xem log
38. Lưu ý quan trọng


---

## Server unreachable

Lỗi ***Server Unreachable*** xảy ra khi người dùng không thể kết nối đến server. Trình duyệt thường hiển thị các thông báo như: *"This site can't be reached"*, *"Connection timed out"* hoặc *"Server not found"*. Điều này có nghĩa là request từ client không thể đến được Web Server, nên hệ thống chưa kịp xử lý bất kỳ logic nào phía backend.




**Nguyên nhân thường gặp**

- ***Server chưa hoạt động:*** Máy chủ có thể đang tắt hoặc service chưa được khởi động
- ***Sai địa chỉ IP hoặc domain:*** Người dùng nhập sai địa chỉ hoặc DNS chưa trỏ đúng
- ***Chưa mở port 80/443:*** Server chưa cho phép kết nối HTTP/HTTPS từ bên ngoài
- ***Firewall chặn kết nối:*** Tường lửa (ufw, iptables, cloud firewall, ...) đang block request
- ***Nginx chưa chạy:*** Không có service nào lắng nghe request từ client
- ***Server không cùng mạng (LAN/WAN):*** Nếu test nội bộ, client và server có thể không cùng subnet

**Cách kiểm tra**

- ***Kiểm tra kết nối mạng***

```
ping <server_ip>
```

Nếu kết nối được có nghĩa là server đang online, nếu không ping được có thể server đang tắt hoặc bị chặn mạng

- ***Kiểm tra trạng thái Nginx***

```
sudo systemctl status nginx
```

Nếu đang active (running) nghĩa là Nginx đang hoạt động. Nếu inactive hoặc failed, có thể Nginx chưa chạy hoặc lỗi

- ***Kiểm tra port có mở không***

```
sudo netstat -tuln | grep 80
```

Xem server có đang lắng nghe port 80 không

***Kiểm tra firewall***

```
sudo ufw status
```

Xem port 80/443 có bị chặn không

**Cách khắc phục**

***Khởi động lại server để đảm bảo máy chủ đang online***

***Khởi động lại Nginx***

```
sudo systemctl start nginx
```

***Mở port HTTP/HTTPS***

```
sudo ufw allow 80

sudo ufw allow 443
```

***Kiểm tra lại IP/domain để đảm bảo truy cập đúng địa chỉ server***

- ***Restart Nginx sau khi cấu hình***

```
sudo systemctl restart nginx
```

# Lỗi 500

Lỗi 500 Internal Server Error xảy ra khi server đã nhận và chuyển request đến backend thành công, nhưng trong quá trình xử lý, ứng dụng gặp lỗi và không thể trả về kết quả hợp lệ. Trong hệ thống sử dụng Nginx và uWSGI, lỗi này thường xuất phát từ phía backend (Django/Flask), không phải do kết nối hay cấu hình mạng.

**Nguyên nhân thường gặp**

- Lỗi code Python (syntax error, NameError, ...)
- Lỗi logic trong xử lý dữ liệu
- Thiếu thư viện hoặc import sai

**Cách kiểm tra**

- Xem log của Nginx

```
sudo tail -f /var/log/nginx/error.log
```

Kết quả nhận được sẽ như sau

```
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ sudo tail -f /var/log/ngi
nx/error.log
2026/04/08 10:45:53 [error] 3109#3109: *1 connect() to unix:/var/www/raspango/raspango.sock
 failed (111: Connection refused) while connecting to upstream, client: 192.168.1.126, serv
er: 192.168.1.168, request: "GET / HTTP/1.1", upstream: "uwsgi://unix:/var/www/raspango/ras
pango.sock:", host: "192.168.1.168"
2026/04/08 13:46:19 [error] 3109#3109: *23 recv() failed (104: Connection reset by peer) wh
ile reading response header from upstream, client: 192.168.1.126, server: 192.168.1.168, re
quest: "GET / HTTP/1.1", upstream: "uwsgi://unix:/var/www/raspango/raspango.sock:", host: "
192.168.1.168"
2026/04/08 13:46:21 [crit] 3109#3109: *21 connect() to unix:/var/www/raspango/raspango.sock
 failed (2: No such file or directory) while connecting to upstream, client: 192.168.1.126,
 server: 192.168.1.168, request: "GET /stream/ HTTP/1.1", upstream: "uwsgi://unix:/var/www/
raspango/raspango.sock:", host: "192.168.1.168", referrer: "http://192.168.1.168/camera/"
2026/04/08 13:50:20 [crit] 3109#3109: *26 connect() to unix:/var/www/raspango/raspango.sock
 failed (2: No such file or directory) while connecting to upstream, client: 192.168.1.126,
 server: 192.168.1.168, request: "GET / HTTP/1.1", upstream: "uwsgi://unix:/var/www/raspang
o/raspango.sock:", host: "192.168.1.168"
2026/04/08 13:58:57 [crit] 3109#3109: *28 connect() to unix:/var/www/raspango/raspango.sock
 failed (2: No such file or directory) while connecting to upstream, client: 192.168.1.126,
 server: 192.168.1.168, request: "GET / HTTP/1.1", upstream: "uwsgi://unix:/var/www/raspang
o/raspango.sock:", host: "192.168.1.168"
2026/04/08 13:59:57 [crit] 3109#3109: *28 connect() to unix:/var/www/raspango/raspango.sock
 failed (2: No such file or directory) while connecting to upstream, client: 192.168.1.126,
 server: 192.168.1.168, request: "GET / HTTP/1.1", upstream: "uwsgi://unix:/var/www/raspang
o/raspango.sock:", host: "192.168.1.168"
```

**Cách khắc phục**

- Kiểm tra lại code backend
- Sửa lỗi logic / biến / import
- Kiểm tra database (nếu có liên quan)

# Lỗi 502

Lỗi ***502 Bad Gateway*** xảy ra khi Nginx đã nhận request từ người dùng và chuyển đến backend (uWSGI/Django/Flask), nhưng không nhận được phản hồi hợp lệ từ backend.

**Nguyên nhân thường gặp**

- uWSGI chưa chạy hoặc bị crash
- Sai đường dẫn socket (.sock) giữa Nginx và uWSGI
- Socket không tồn tại hoặc không đúng path
- Sai quyền truy cập socket (permission denied)
- Backend bị treo hoặc không phản hồi
- Lỗi kết nối giữa Nginx và uWSGI

**Cách kiểm tra**

- Kiểm tra uWSGI có đang chạy không

```
ps aux | grep uwsgi
```

  Nếu không thấy process nghĩa là uWSGI chưa chạy hoặc đã bị crash
- Kiểm tra file socket

```
ls -l /var/www/raspango/raspango.sock
```

  Nếu không tồn tại nghĩa là uWSGI chưa tạo socket hoặc sai config
- Xem log uWSGI / backend

```
sudo vim /var/www/raspango/uwsgi.log
```

**Cách khắc phục**

- Chạy lại uWSGI

```
sudo systemctl restart uwsgi
```

- Kiểm tra và sửa lại đường dẫn socket để đảm bảo giống nhau giữa Nginx và uWSGI
- Kiểm tra quyền socket

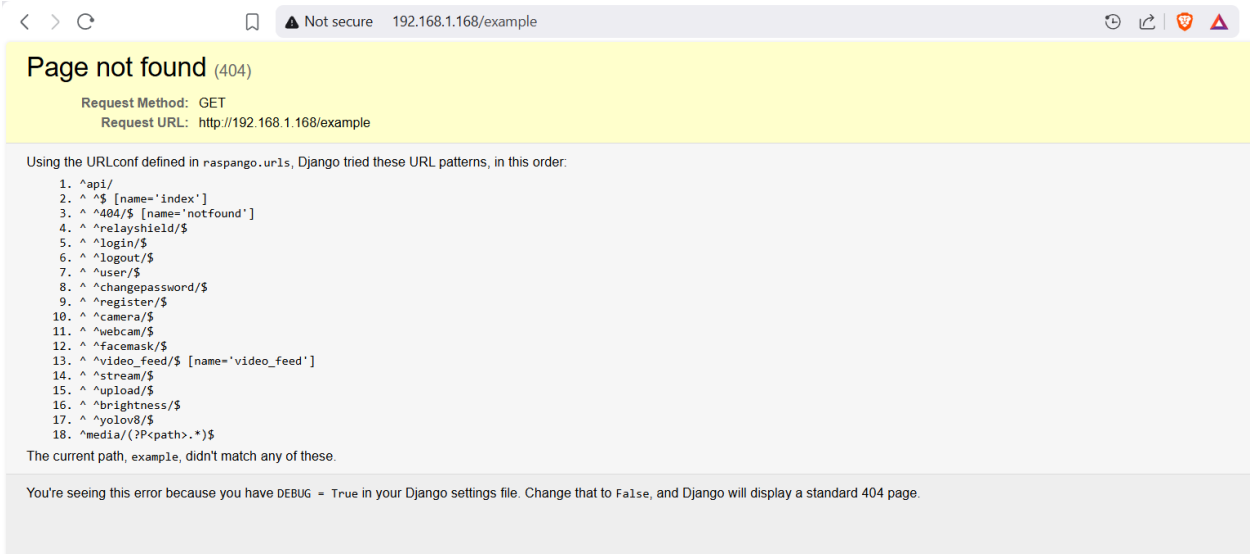
```
chmod 666 /var/www/raspango/raspango.sock
```

- Restart Nginx

```
sudo systemctl restart nginx
```

# Lỗi 404

Lỗi 404 Not Found xảy ra khi server đã nhận được request từ người dùng nhưng không tìm thấy tài nguyên tương ứng với URL được yêu cầu, chẳng hạn như trang web, API hoặc file. Nguyên nhân thường do URL nhập sai, route chưa được khai báo, hoặc tài nguyên đã bị xóa hoặc di chuyển, nên server không thể trả về nội dung phù hợp.




### Nguyên nhân thường gặp

- Sai URL (gõ nhầm đường dẫn)
- Route trong backend (Django/Flask) chưa được khai báo
- URL không khớp với cấu hình routing
- File tĩnh (static/media) không tồn tại
- Sai cấu hình Nginx cho static/media
- Sai server_name hoặc location trong Nginx

**Cách kiểm tra**

- Kiểm tra URL truy cập để đảm bảo đường dẫn đúng chính tả, đúng endpoint đã được backend định nghĩa
- Kiểm tra route trong backend
- Kiểm tra cấu hình Nginx để đảm bảo trỏ đúng thư mục và không sai path
- Kiểm tra file tĩnh

```
ls -l /var/www/raspango/code/server/web/static
```

- Xem log Nginx

```
sudo tail -f /var/log/nginx/error.log
```

**Cách khắc phục**

- Kiểm tra và sửa lại URL truy cập
- Khai báo thêm route trong backend
- Kiểm tra lại cấu hình routing (Django/Flask)
- Sửa cấu hình Nginx cho đúng location
- Đảm bảo file static/media tồn tại
- Reload lại Nginx sau khi chỉnh sửa:

```
sudo systemctl reload nginx
```

## Cách xem log

Trong quá trình vận hành hệ thống web sử dụng Nginx và uWSGI, việc xem log là bước quan trọng để xác định nguyên nhân lỗi. Log sẽ ghi lại các thông tin như request từ người dùng, lỗi kết nối, lỗi backend hoặc lỗi cấu hình.

**Xem log của Nginx**

Nginx lưu log tại hai file chính:

- **Access log**: ghi lại các request truy cập
- **Error log**: ghi lại các lỗi phát sinh

Cách xem:

```
sudo tail -f /var/log/nginx/access.log

sudo tail -f /var/log/nginx/error.log
```

Ý nghĩa :

- **access.log:** biết được người dùng đã truy cập URL nào, status code là gì (200, 404, 503,...)
- **error.log:** hiển thị các lỗi như không kết nối được backend, timeout, permission denied, socket không tồn tại,...

```
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ sudo tail -f /var/log/ngi
nx/access.log
192.168.1.126 - - [08/Apr/2026:15:04:38 +0700] "GET /stream/ HTTP/1.1" 200 112323 "http://1
92.168.1.168/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/146.0.0.0 Safari/537.36"
192.168.1.126 - - [08/Apr/2026:15:04:39 +0700] "GET / HTTP/1.1" 502 568 "-" "Mozilla/5.0 (W
indows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 Safari/
537.36"
192.168.1.126 - - [08/Apr/2026:15:05:19 +0700] "GET / HTTP/1.1" 200 3833 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 Safari
/537.36"
192.168.1.126 - - [08/Apr/2026:15:05:21 +0700] "GET / HTTP/1.1" 200 3833 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 Safari
/537.36"
192.168.1.126 - - [08/Apr/2026:15:05:30 +0700] "GET / HTTP/1.1" 200 3833 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 Safari
/537.36"
192.168.1.126 - - [08/Apr/2026:15:05:40 +0700] "GET /404 HTTP/1.1" 301 0 "-" "Mozilla/5.0 (
Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 Safari
/537.36"
192.168.1.126 - - [08/Apr/2026:15:05:40 +0700] "GET /404/ HTTP/1.1" 500 59901 "-" "Mozilla/
5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 S
afari/537.36"
192.168.1.126 - - [08/Apr/2026:15:06:00 +0700] "GET /example HTTP/1.1" 404 1177 "-" "Mozill
a/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0
 Safari/537.36"
192.168.1.126 - - [08/Apr/2026:15:21:17 +0700] "GET /stream/ HTTP/1.1" 200 100368 "http://1
92.168.1.168/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/146.0.0.0 Safari/537.36"
192.168.1.126 - - [08/Apr/2026:15:22:35 +0700] "GET /stream/ HTTP/1.1" 200 141408 "http://1
92.168.1.168/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
ecko) Chrome/146.0.0.0 Safari/537.36"
```

### Xem log của uWSGI

uWSGI có thể ghi log ra terminal hoặc lưu vào file log tùy theo cấu hình trong file .ini

Trong trường hợp cấu hình trong file raspango.ini có sử dụng:

```
logto = uwsgi.log
```

hoặc

```
logto = /var/www/raspango/uwsgi.log
```

Khi đó, toàn bộ log của uWSGI sẽ được ghi trực tiếp vào file uwsgi.log trong thư mục dự án.

```
tail -f uwsgi.log
```

```
(venv) ubuntu@ubuntu-HP-Z230-Tower-Workstation:/var/www/raspango$ tail -f uwsgi.log
Not Found: /example
[pid: 4679|app: 0|req: 3/6] 192.168.1.126 () {44 vars in 960 bytes} [Wed Apr  8 08:06:00 20
26] GET /example => generated 4660 bytes in 3 msecs (HTTP/1.1 404) 4 headers in 116 bytes (
1 switches on core 0)
Wed Apr  8 08:21:17 2026 - SIGPIPE: writing to a closed pipe/socket/fd (probably the client
 disconnected) on request /stream/ (ip 192.168.1.126) !!!
Wed Apr  8 08:21:17 2026 - uwsgi_response_write_body_do(): Broken pipe [core/writer.c line
429] during GET /stream/ (192.168.1.126)
OSError: write error
[pid: 4682|app: 0|req: 1/7] 192.168.1.126 () {46 vars in 913 bytes} [Wed Apr  8 08:05:19 20
26] GET /stream/ => generated 591108 bytes in 958672 msecs (HTTP/1.1 200) 3 headers in 95 b
ytes (9534 switches on core 0)
Wed Apr  8 08:22:35 2026 - SIGPIPE: writing to a closed pipe/socket/fd (probably the client
 disconnected) on request /stream/ (ip 192.168.1.126) !!!
Wed Apr  8 08:22:35 2026 - uwsgi_response_write_body_do(): Broken pipe [core/writer.c line
429] during GET /stream/ (192.168.1.126)
OSError: write error
[pid: 4680|app: 0|req: 3/8] 192.168.1.126 () {46 vars in 913 bytes} [Wed Apr  8 08:05:30 20
26] GET /stream/ => generated 632400 bytes in 1025362 msecs (HTTP/1.1 200) 3 headers in 95
bytes (10200 switches on core 0)
```

**Trong log uWSGI có thể thấy:**

- Lỗi import module Python
- Lỗi Django
- Lỗi database
- request xử lý thành công hoặc thất bại

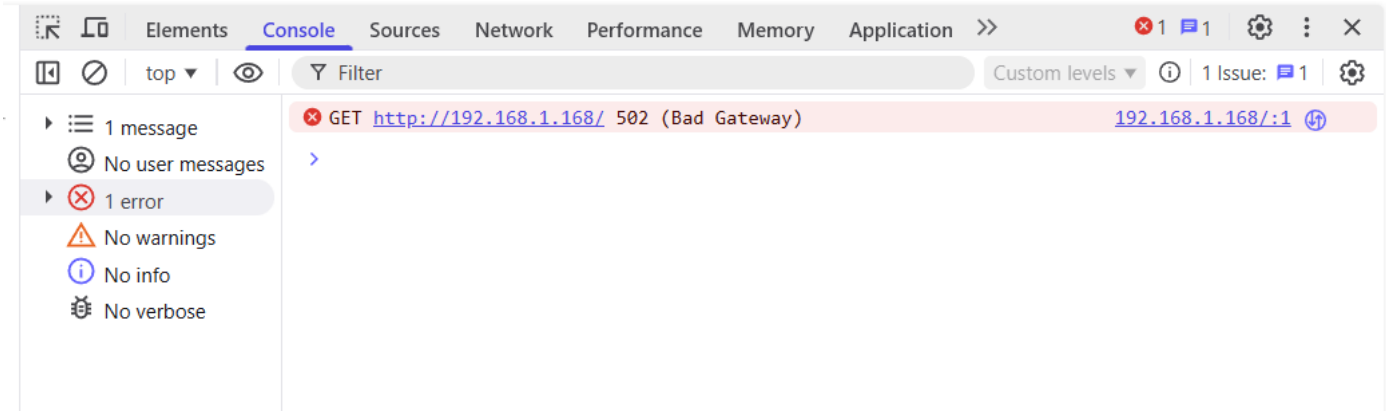
**Xem log thông qua trình duyệt**

Ngoài việc xem log trực tiếp trên server bằng terminal, người dùng cũng có thể kiểm tra kết quả và trạng thái hệ thống thông qua trình duyệt web. Khi truy cập vào ứng dụng web bằng trình duyệt, các request sẽ được gửi đến server và phản hồi lại với các mã trạng thái HTTP như:

- **200 OK**: request thành công
- **404 Not Found**: không tìm thấy tài nguyên
- **500 Internal Server Error**: lỗi phía backend

- **502 Bad Gateway / 504 Gateway Timeout**: lỗi kết nối giữa Nginx và uWSGI

Để xem log trên server, trình duyệt web cũng cung cấp công cụ Developer Tools (mở bằng phím F12) để kiểm tra và debug hoạt động của website (xem log tại mục console).




## Lưu ý quan trọng

**Không nâng cấp Python!**

Trong quá trình triển khai và vận hành hệ thống web sử dụng Django và uWSGI, không nên tự ý nâng cấp phiên bản Python trên server nếu không có kế hoạch kiểm thử rõ ràng.

**Việc nâng cấp Python có thể dẫn đến một số vấn đề như:**

- Không tương thích với các thư viện đã cài đặt (Django, uWSGI, numpy, pandas, ...)
- uWSGI có thể không hoạt động nếu không được build lại với phiên bản Python mới
- Các package trong virtual environment (venv) có thể bị lỗi hoặc không chạy được
- Ứng dụng đang hoạt động ổn định có thể phát sinh lỗi không mong muốn

**Vì vậy, khi hệ thống đã được triển khai ổn định:**

- Nên giữ nguyên phiên bản Python đang sử dụng
- Chỉ nâng cấp khi thật sự cần thiết và đã kiểm tra tương thích
- Nếu nâng cấp, cần tạo lại môi trường ảo (venv) và cài đặt lại toàn bộ dependencies

---

# Chương 8: Checklist triển khai hệ thống

Chương này hướng dẫn cách sử dụng checklist để nghiệm thu hệ thống web server sau khi hoàn tất quá trình cài đặt và triển khai. Người học cần thực hiện lần lượt từng bước kiểm tra theo danh sách đã cung cấp, đảm bảo mỗi hạng mục đều đạt yêu cầu trước khi chuyển sang bước tiếp theo.

Trong quá trình thực hiện, nếu một mục kiểm tra không đạt, cần dừng lại và xử lý lỗi tương ứng trước khi tiếp tục. Checklist này được thiết kế để giúp người học nhanh chóng xác định tình trạng hệ thống, đảm bảo server hoạt động ổn định và sẵn sàng đưa vào sử dụng thực tế.

## Nội dung


39. Kết nối server
40. Firewall
41. Domain
42. Web Server (Nginx)
43. Backend Service
44. Kiểm tra truy cập qua IP
45. Kiểm tra truy cập qua Domain
46. Tự động khởi động
47. Log hệ thống
48. Quyền file
49. Kiểm tra deploy lại


---

## Kết nối server

- ☐ **SSH vào server thành công**

```
ssh ubuntu@<server_ip>
```

- ☐ **Không sử dụng password (chỉ dùng SSH key)**

Nếu login mà không bị hỏi password thì được

## Firewall

- ☐ **Firewall đã bật**

```
sudo ufw status
```

Trạng thái *"Status: active"* là được

- ☐ **Port 22 (SSH) đã mở**
- ☐ **Port 80 (HTTP) đã mở**

```
sudo ufw status
```

Có dòng

```
22/tcp ALLOW

80/tcp ALLOW
```

## Domain

- ☐ **Domain trỏ đúng IP server**

```
ping yourdomain.com
```

- ☐ **Ping domain trả về đúng IP**

So sánh với IP server

## Web Server (Nginx)

☐ **Nginx đang chạy**

```
systemctl status nginx
```

Trạng thái active là được

☐ **Không báo lỗi khi kiểm tra service**

```
nginx -t
```

Kiểm tra syntax của nginx

## Backend Service

☐ **Service backend (uwsgi/gunicorn) đang chạy**

```
systemctl status uwsgi
```

☐ **Không bị crash hoặc restart liên tục**
Không thấy failed hoặc restarting

## Kiểm tra truy cập qua IP

☐ **Truy cập được qua địa chỉ IP server**

```
curl http://<server_ip>
```

☐ **Không xuất hiện lỗi 502 / 503**
Không thấy “Bad Gateway” / “Service Unavailable”

## Kiểm tra truy cập qua Domain

☐ **Truy cập domain hiển thị website đúng**

```
curl http://yourdomain.com
```

☐ **Không lỗi 404**

Không thấy “404 Not Found”

## Tự động khởi động

☐ **Nginx tự chạy khi reboot**

```
systemctl is-enabled nginx
```

☐ **Backend service tự chạy khi reboot**

```
systemctl is-enabled uwsgi
```

## Log hệ thống

☐ **Log Nginx không có lỗi nghiêm trọng**

```
tail -n 20 /var/log/nginx/error.log
```

☐ **Log backend không có lỗi nghiêm trọng**

```
journalctl -u <service_name> -n 20
```

## Quyền file

☐ **File thuộc đúng user**

```
ls -l
```

☐ **Không gặp lỗi permission denied**
Truy cập web không lỗi 403

## Kiểm tra deploy lại

☐ **Restart service thành công**

```
systemctl restart nginx

systemctl restart uwsgi
```

☐ **Website vẫn hoạt động sau khi restart**
curl http://yourdomain.com