KHÓA HỌC FRONT-END

Buổi 13: Học GIT, GITHUB

Nội dung

01 Học GIT

02 Sử dụng GITHUB

03 Deploy code lên Vercel

01. Học GIT

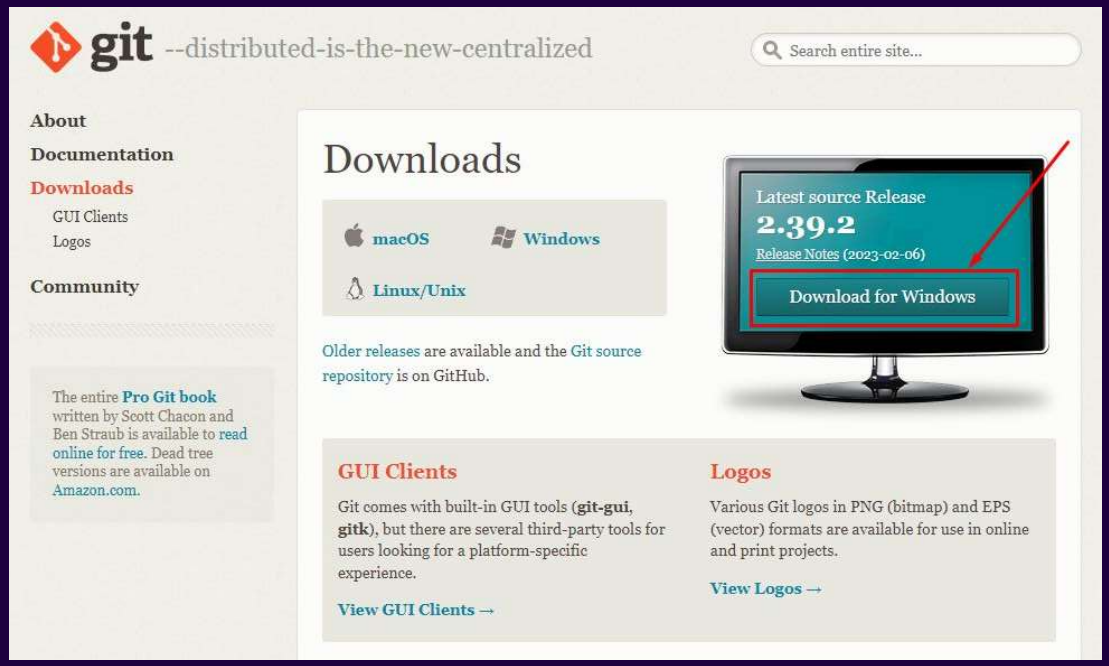
1.1. Giới thiệu chung

- Git là **một hệ thống quản lý phiên bản phân tán** (Distributed Version Control System – DVCS).
- Git cung cấp cho mỗi lập trình viên **kho lưu trữ** (repository) riêng **chứa toàn bộ lịch sử thay đổi**.
- Ưu điểm: tốc độ **nhanh**, **đơn giản**, phân tán, phù hợp với dự án lớn nhỏ.
- Git là một trong những hệ thống quản lý phiên bản phân tán **phổ biến nhất hiện nay**.

01. Học GIT

1.2. Cài đặt GIT

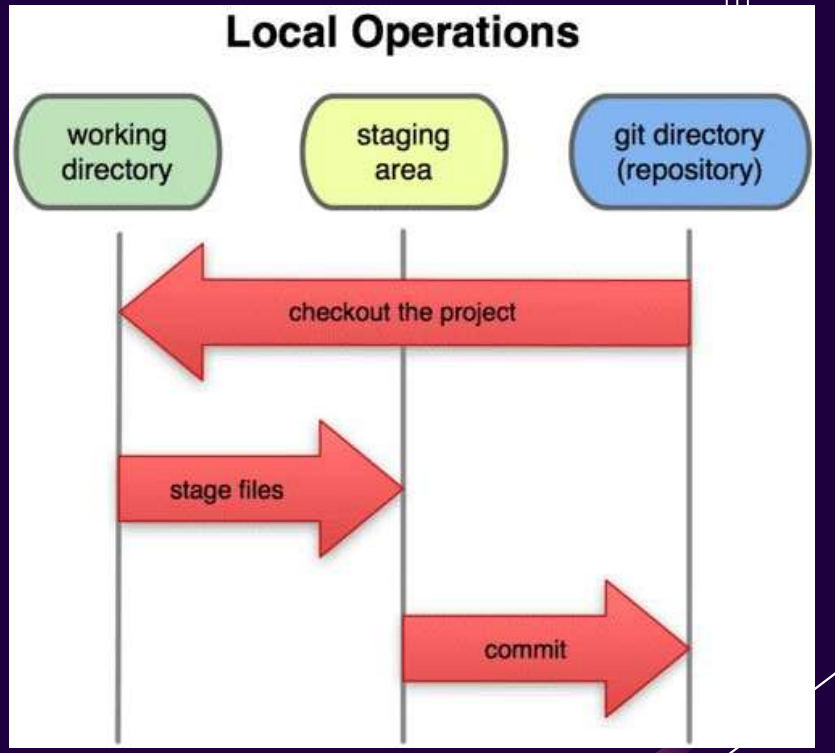
- https://git-scm.com/downloads



01. Học GIT

1.3. Các thuật ngữ: Working directory, Staging area, Repository

- **Working directory** (Thư mục làm việc): Khu vực chứa dự án mà chúng ta đang làm việc.
- **Staging area** (Khu vực sắp xếp): khu vực chứa thông tin thay đổi của các file.
- **Repository** (Git directory - thư mục git): **Kho lưu trữ** để lưu trữ dữ liệu, lịch sử các phiên bản.



01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

- **git –version**: Xem phiên bản hiện tại đang được cài trên máy tính.
- **git –help**: Xem danh sách các câu lệnh Git.
- **git config**: Để đặt username và email của bạn trong file config của git.
    - Để xem thông tin cấu hình username: **git config –global user.name**
    - Để xem thông tin cấu hình email: **git config –global user.email**
    - Để đặt username mới: **git config –global user.name "Đặng Phương Nam"**
    - Để đặt email mới: **git config –global user.email "namdp.1999@gmail.com"**

01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

- **git init**
  - **Khởi tạo** 1 git **repository** (kho lưu trữ) trong một **dự án mới** hoặc **dự án đã có** (Dùng trong thư mục gốc của dự án).
  - Khi khởi tạo xong, **trong thư mục gốc** (dự án đang làm) sẽ sinh ra một thư mục tên là **.git**, thư mục này sẽ **lưu toàn bộ thông tin lịch sử** của dự án mà bạn làm. Chúng ta không cần quan tâm bên trong thư mục này chứa gì.
- **git status**
  - Để **xem trạng thái** của những **file đã được thay đổi** (bao gồm: thêm, sửa, xóa) trong dự án.
  - Ví dụ: tạo file README.md trong dự án (file này để giới thiệu và hướng dẫn cách cài đặt dự án). Sau đó gõ lệnh git status, ta sẽ thấy có file README.md được thêm mới.

01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

- **git add ten_file**
  - Thông tin của các file sẽ được lưu vào **Staging area**.
  - Staging area có tác dụng **sắp xếp lại** nhưng file đã add vào.
  - Ví dụ: tạo ra các file rồi thử lại câu lệnh này.
- **git add .**
  - Thêm thay đổi cho tất cả các file.
  - Ví dụ: tạo ra các file rồi thử lại câu lệnh này.

01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

- **git commit -m "Nội dung..."**
  - Để đưa những file ở vùng Staging area chuyển sang Repository, mục đích là tạo ra 1 phiên bản mới vào lưu vào lịch sử của Repository.
  - Với điều kiện các tập tin, thư mục được thay đổi **đã phải nằm trong Staging Area**.

01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

- **Nhắc lại quy trình:**
  - Khi **thêm, sửa, xóa file** thì vẫn đang ở **Working directory** *(Chạy lệnh git status, những file **màu đỏ** là đang nằm trong **Working directory**).*
  - Sau đó chạy lệnh **git add ten_file** thì thông tin của các file sẽ được lưu vào **Staging area**, Staging area có tác dụng sắp xếp lại nhưng file đã add vào *(Giả sử có 1 file được thay đổi, ta add vào lần 1, sau đó ta lại sửa file đó và add vào lần 2, thì Staging area chỉ lấy lần 2. Chạy lệnh git status, những file **màu xanh lá** là đang nằm trong **Staging area**).*
  - Sau đó chạy lệnh **git commit -m "Nội dung commit..."**. Lúc này, những file đã được đánh dấu ở vùng Staging area sẽ được lưu vào **Repository**.

01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

- **git log**: Giúp bạn **xem lại thông tin lịch** sự commit, nhằm giám sát sự thay đổi của dự án. Commit mới sẽ hiện bên trên, commit cũ sẽ hiện bên dưới *(Nếu gặp chữ **END** thì nhấn phím **q** để thoát)*.
- **git show commit_id**: Dùng để **xem chi tiết một commit**
- **git diff**: Xem sự thay đổi của một file sau khi chúng ta chỉnh sửa (File đó vẫn đang ở khu vực Working directory).
- **gitk**: Mở dashboard xem trực quan hơn.

01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

- **git checkout – ten_file**: Bỏ đi những thay đổi của file, để file đó trở về như lúc ban đầu. Áp dụng cho file đang ở vùng Working directory.
- **git reset HEAD ten_file** hoặc **git reset ten_file**: Chuyển file đó từ vùng Staging area trở lại vùng Working directory.

01. Học GIT

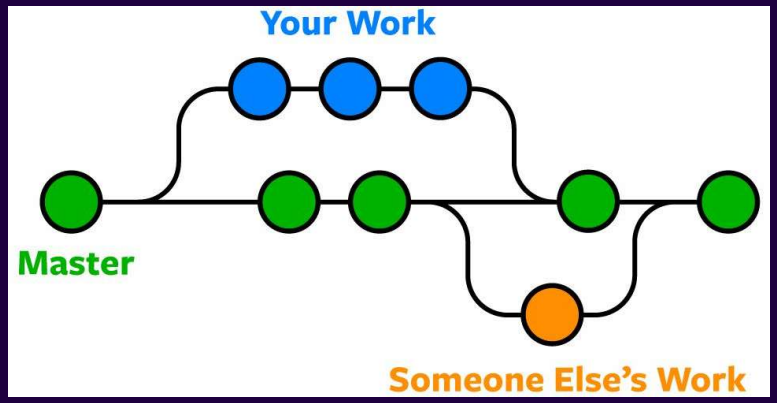
1.4. Các câu lệnh GIT sử dụng phổ biến

- **git reset –soft commit_id**
  - Dùng để chuyển từ trạng thái đã commit về trạng thái trước lúc chạy lệnh git commit.
  - Tức là **từ Repository về lại Staging area**. Trong đó **commit_id** là mã của nhánh mà ta muốn quay lại.
- **git reset –mixed commit_id**
  - Dùng để chuyển từ trạng thái đã commit về trạng thái trước lúc chạy lệnh git add.
  - Tức là **từ Repository về lại Working directory**. Trong đó **commit_id** là mã của nhánh mà ta muốn quay lại.

01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

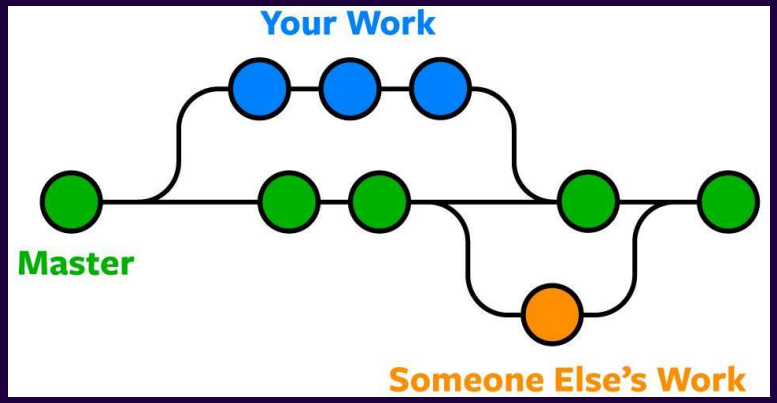
- **git branch**: Xem danh sách các nhánh. Các branch (nhánh) đại diện cho các phiên bản cụ thể của một kho lưu trữ tách ra từ project chính của bạn.
    - Nhánh master là nhánh chính, để sau này deploy lên server.
- **git checkout -b ten_nhanh**: Tạo một nhánh mới và chuyển sang nhánh đó.
- **git checkout ten_nhanh**: Chuyển sang nhánh khác.



01. Học GIT

1.4. Các câu lệnh GIT sử dụng phổ biến

- **git merge ten_nhanh**: Để merge nhánh <ten_nhanh> vào trong nhánh hiện tại.
    - Ví dụ: Ta có 2 nhánh A và B, để hợp nhất nhánh B vào trong nhánh A ta làm như sau: Dùng lệnh **git checkout A** để chuyển sang nhánh A, sau đó chạy lệnh **git merge B** để hợp nhất nhánh B vào nhánh A.
- **git branch -D ten_nhanh**: Để xóa nhánh.



02. Sử dụng GITHUB

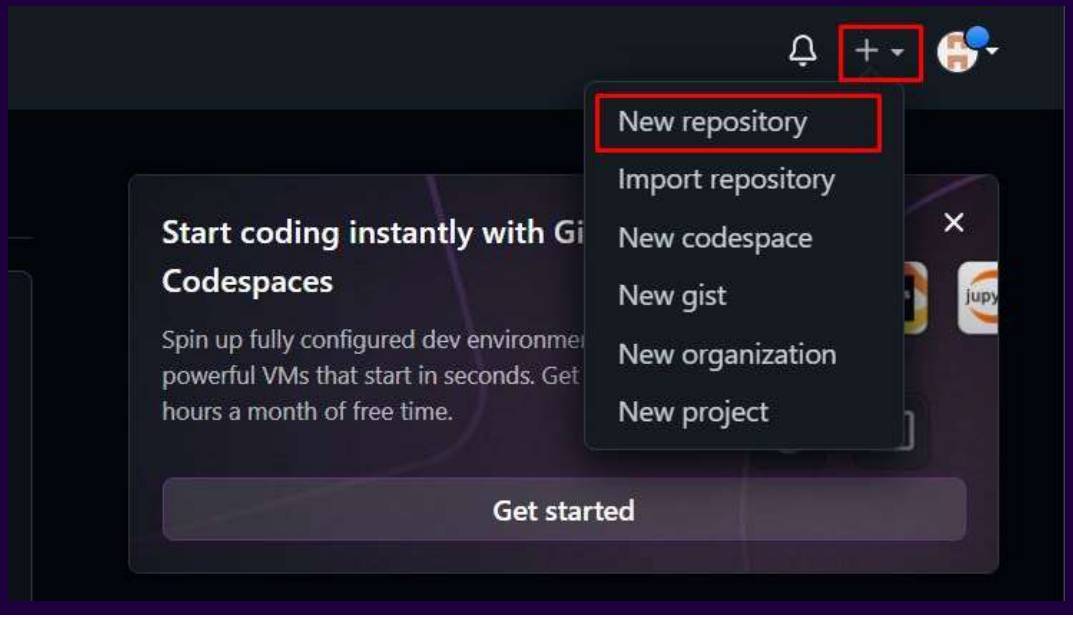
2.1. Giới thiệu chung

- GitHub là một hệ thống quản lý dự án và phiên bản code.
- GitHub là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn Git cho các dự án phần mềm. Github có đầy đủ những tính năng của Git.
- Giúp đồng bộ source code của team lên 1 server.

02. Sử dụng GITHUB

2.2. Tạo tài khoản và tạo Repository

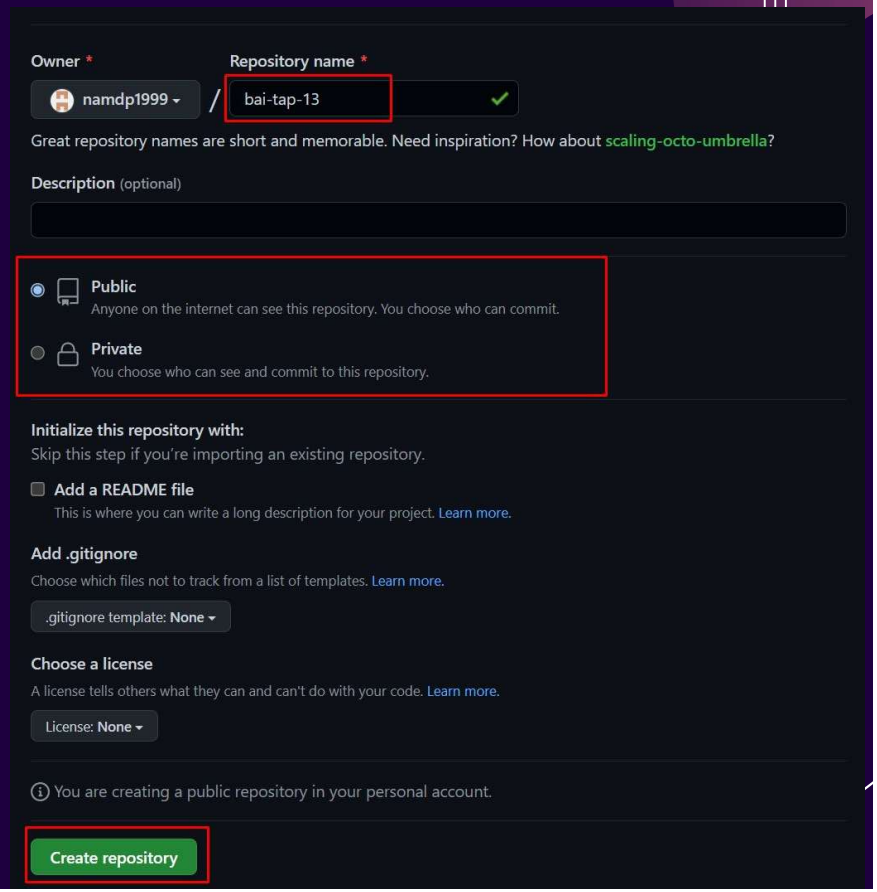
- Link website: https://github.com/
- Cách tạo mới một Repository



02. Sử dụng GITHUB

2.2. Tạo tài khoản và tạo Repository

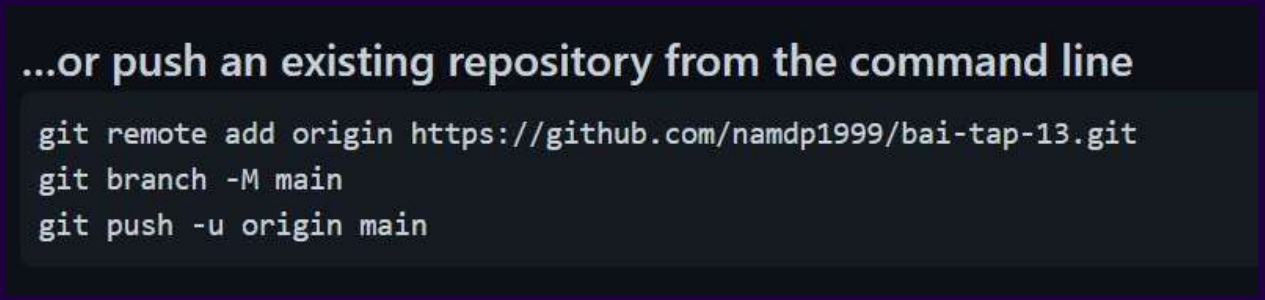
- Link website: https://github.com/
- Cách tạo mới một Repository



02. Sử dụng GITHUB

2.3. Đẩy code lần đầu lên GITHUB khi project đã có GIT

- Bước 1: **git remote add origin url_github_https**
- Bước 2: **git branch -M main**
- Bước 3: **git push -u origin main**



02. Sử dụng GITHUB

2.4. Đẩy code lần đầu lên GITHUB khi project chưa có GIT

- Bước 1: **git init**
- Bước 2: **git add .**
- Bước 3: **git commit -m "Nội dung commit"**
- Bước 4: **git branch -M main**
- Bước 5: **git remote add origin url_github_https**
- Bước 6: **git push -u origin main**

02. Sử dụng GITHUB

2.5. Đẩy code lên GITHUB cho các lần tiếp theo

- Bước 1: **git add .**
- Bước 2: **git commit -m "Nội dung commit"**
- Bước 3: **git push**

2.6. Kéo code từ GITHUB về máy

- Bước 1: **git pull origin ten_nhanh**

02. Sử dụng GITHUB

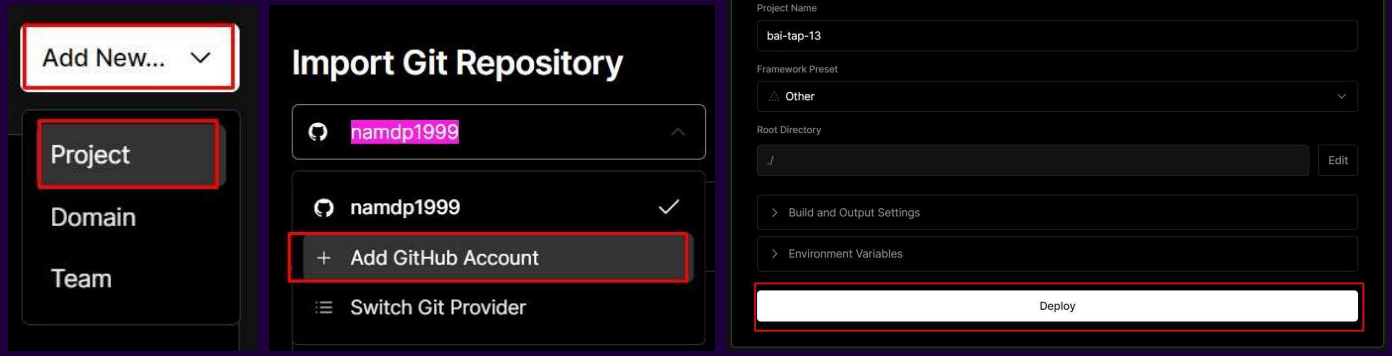
2.7. Lời khuyên khi thao tác thường xuyên với Git trong công việc

- **Nên commit thường xuyên**: Tách nhỏ commit của bạn và commit thường xuyên nhất có thể. Điều này giúp các thành viên trong nhóm dễ dàng tích hợp công việc của họ hơn mà không gặp phải xung đột khi hợp nhất code.
- **Test trước rồi mới commit**: Không bao giờ commit nếu chưa hoàn tất một công việc. Cần phải test các thay đổi của bạn trước khi chia sẻ chúng với người khác.
- **Viết ghi chú khi commit**: Viết ghi chú khi commit để cho các thành viên khác trong nhóm biết loại thay đổi bạn đã thực hiện. Hãy mô tả rõ ràng.

03. Deploy code lên Vercel

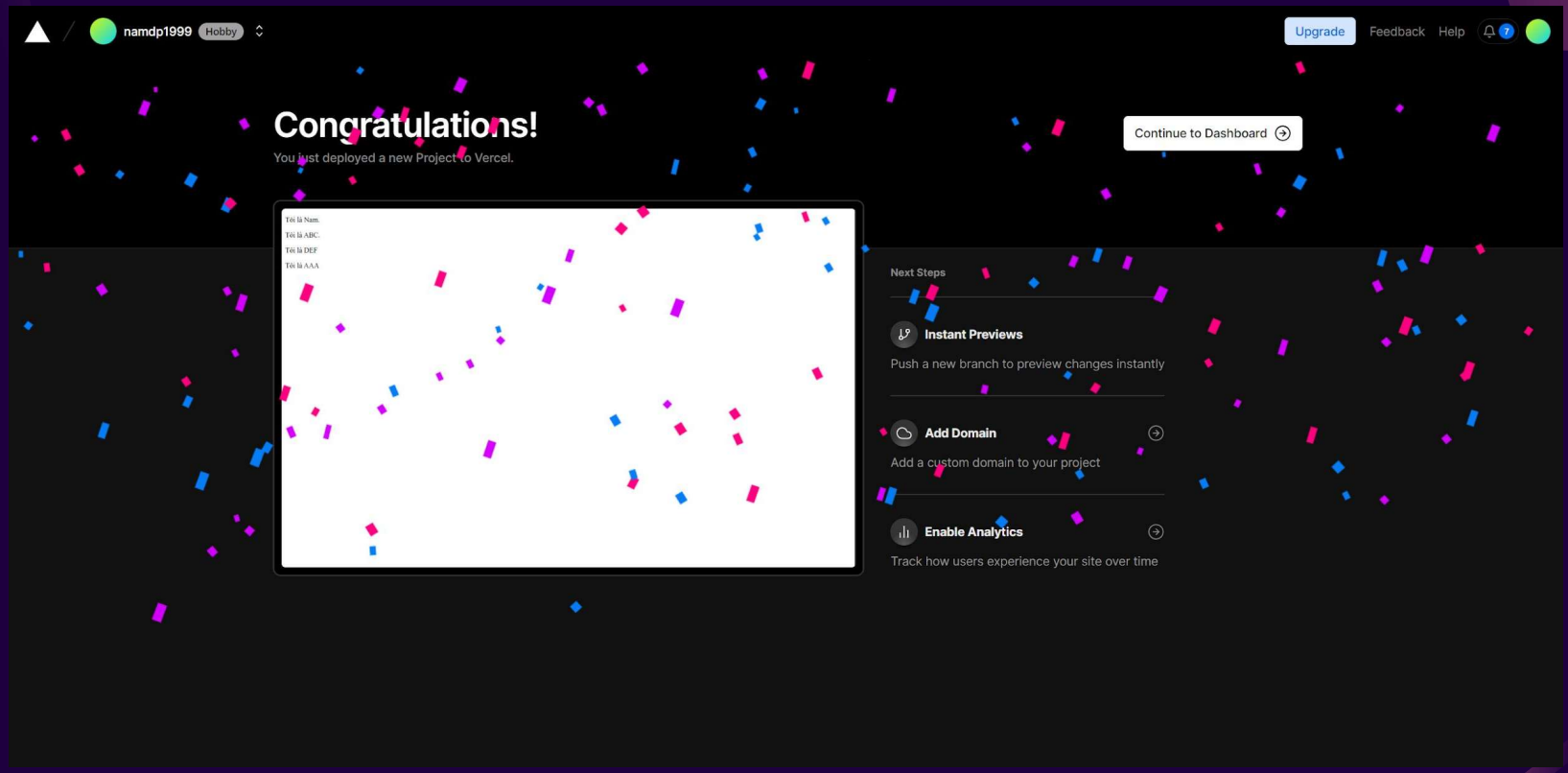
Hướng dẫn deploy (triển khai) code lên Vercel

- Link website: https://vercel.com/
- Các bước deploy code:
  - Bước 1: Chọn "Add New..."
  - Bước 2: Chọn "Project"
  - Bước 3: Chọn "Add Github Account" (nếu chưa có)
  - Bước 4: Bấm Deploy



03. Deploy code lên Vercel

Hướng dẫn deploy (triển khai) code lên Vercel



Bài tập

- Câu 1: Đẩy hết các bài tập sau lên GITHUB, mỗi bài tập là một project riêng biệt (tức là mỗi bài là một Repository)
  - Bài 1
  - Bài 2
  - Bài 3
  - Bài 4
  - Bài 5
  - Bài 6
  - Project mini 1
- Câu 2: Deploy các project đó lên trên Vercel
- Câu 3: Chèn link bài tập (link Vercel) vào bảng Excel sau: