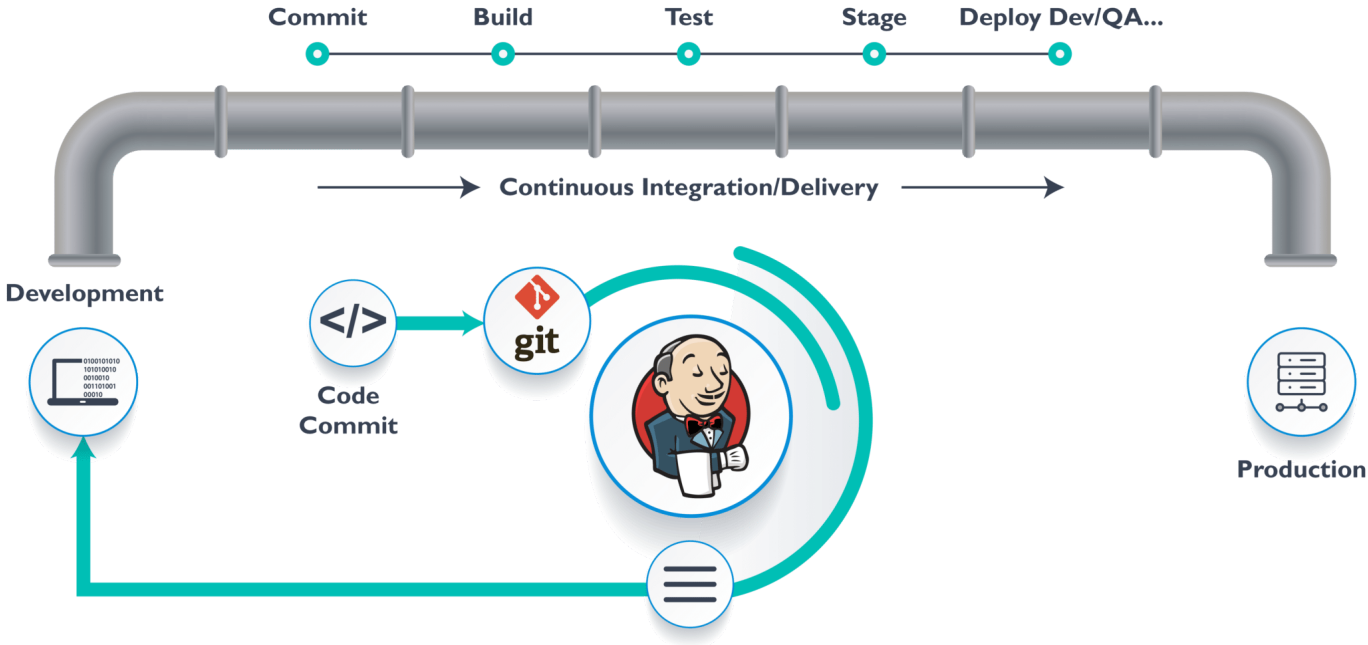
Tìm hiểu quy trình Continuous Integation với Jenkins

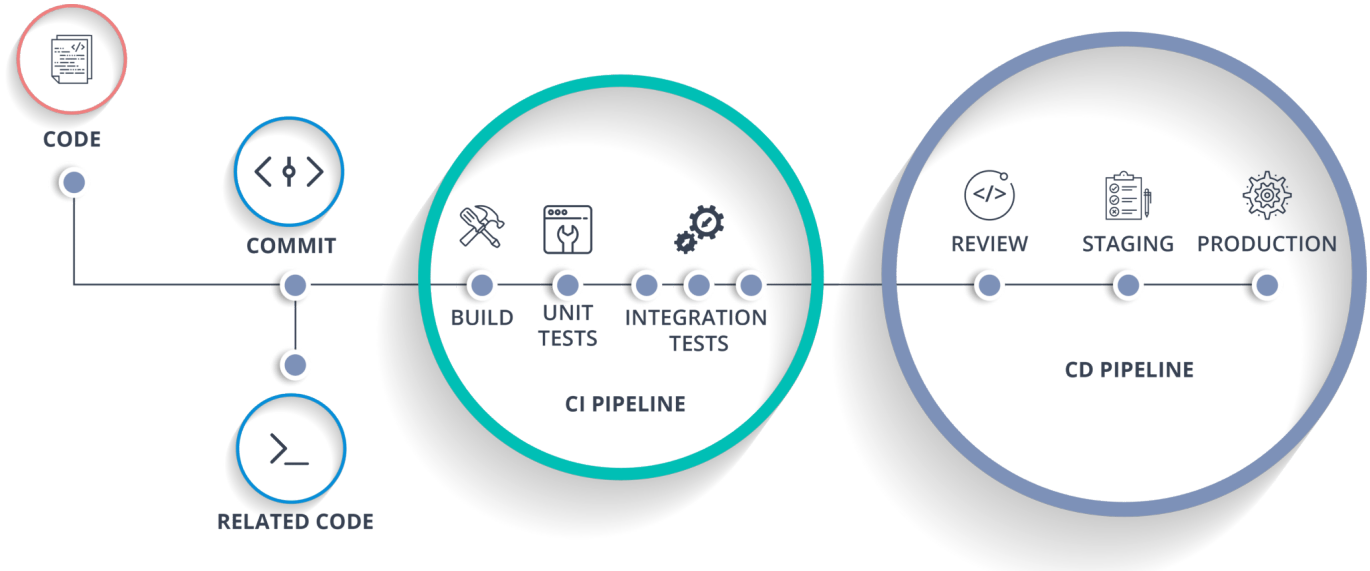
1. Jenkins là gì ?

1.1. Khái niệm

Jenkins là một opensource dùng để thực hiện chức năng tích hợp liên tục (gọi là **CI – Continuous Integration**) và xây dựng các tác vụ tự động hóa. Nó tích hợp các source code của các members trong team lại nhanh chóng một cách liên tục, theo dõi sự thực thi và trạng thái thông qua các bước kiểm thử (**Integration test**, **units test**). Tất nhiên là nhằm giúp sản phẩm chạy ổn định.



Quy trình CICD của Jenkins:



1.2. Chu trình làm việc

1. Bước đầu tiên, các thành viên trong team dev sẽ bắt đầu pull code mới nhất từ repo về branch để thực hiện các yêu cầu chức năng nhất định.

2. Tiếp đó là quá trình lập trình và test code để đảm bảo chất lượng của chức năng cũng như toàn bộ source code.
3. Thành viên code xong thì sẵn sàng cho việc commit vào branch develop của team.
4. Thành viên cập nhật code mới từ repo về local repo
5. Merge code và giải quyết conflict.
6. Build và đảm bảo code pass qua các tests dưới local.
7. Commit code lên repo
8. Máy chủ CI lắng nghe các thay đổi code từ repository và có thể tự động build/test, sau đó đưa ra các thông báo (pass/failure) cho các thành viên.

2. Cài đặt Jenkins

2.1. Chuẩn bị

- Chuẩn bị máy ảo chạy hệ điều hành Windows hoặc Linux cấu hình tối thiểu: 2 CPU, 2GB RAM, >= 50GB DISK.
- Cài đặt trước docker trên máy.

2.2. Cài đặt

Bước 1: Chỉnh sửa cấu hình network docker để có thể chạy docker trong docker (trong quá trình build Jenkins sẽ gọi đến docker của host thông qua kết nối socket để tạo docker phục vụ build code). Chỉnh sửa nội dung file `/lib/systemd/system/docker.service`. Chỉnh sửa nội dung như cấu hình sau đây:

```
ExecStart=/usr/bin/dockerd -H unix://var/run/docker.sock -H
tcp://172.16.87.131 --containerd=/run/containerd/containerd.sock
```

với giao thức TCP sẽ chọn địa chỉ IP là IP của máy **Host**.

Bước 2: Tạo user Jenkins (tránh việc hạy Jenkins bằng Root).

```
# create new user
$ useradd -m -d /home/Jenkins -s /bin/bash Jenkins
# add Jenkins user to docker group
$ usermod -aG docker Jenkins

$ su - Jenkins
```

Bước 3: Chạy Jenkins bằng docker.

- Tạo thư mục backup dữ liệu cho Jenkins.

```
$ mkdir data
```

- Lấy UserId của user jenkins:

```
$ id
uid=1001(jenkins) gid=1001(jenkins) groups=1001(jenkins),999(docker)
```

- Khởi tạo docker container.

```
$ docker run -v /var/run/docker.sock:/var/run/docker.sock -v $(which
docker):$(which docker) -v `PWD`/data:/var/jenkins_home -p 8080:8080 --user
1001:999 --name jenkins-server -d docker pull jenkins/jenkins
```

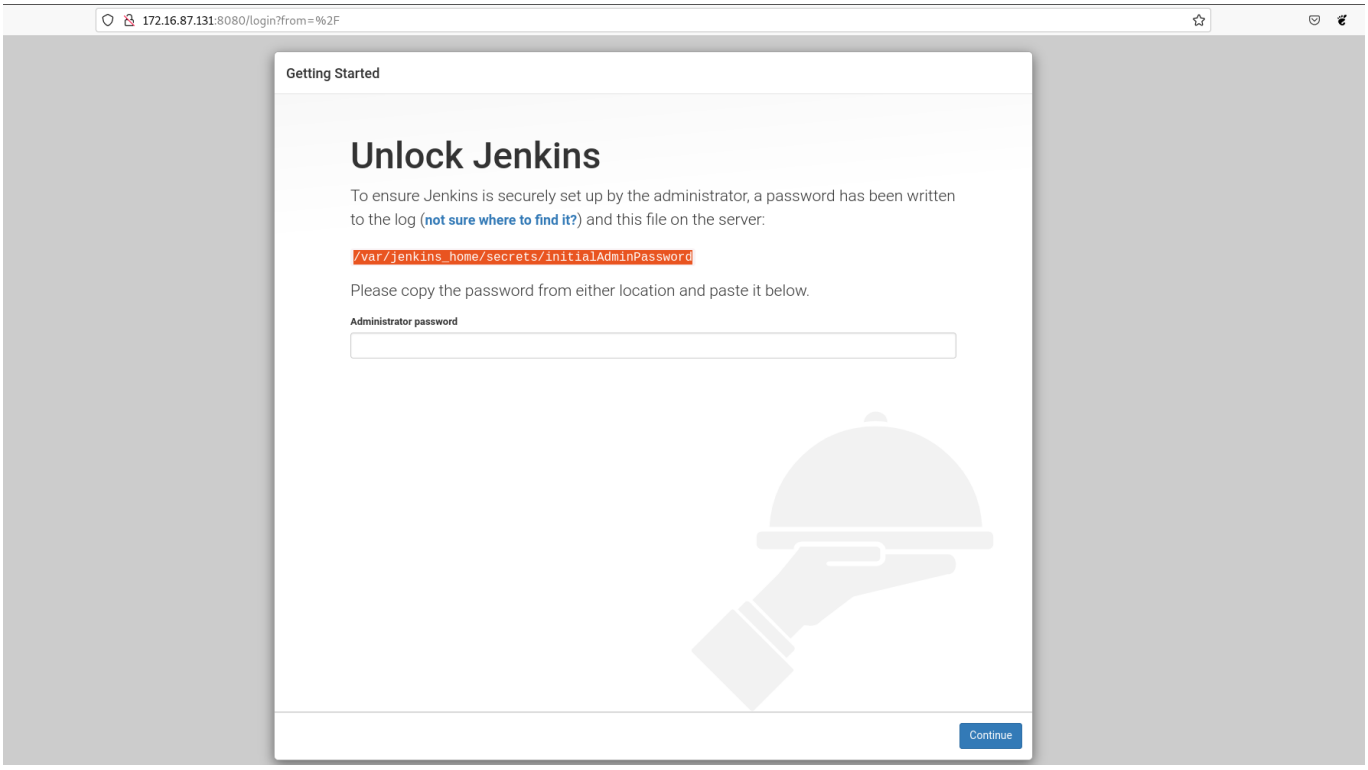
```
jenkins@ip-172-31-17-212:~$ docker container ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED        STATUS          PORTS                                                  NAMES
4a530855b0de   jenkins/jenkins:lts   "/sbin/tini -- /usr/…"   14 hours ago   Up 29 seconds   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 50000/tcp   jenkins-server
```

- Sau khi chạy ổn định truy cập vào `http://localhost:8080` và lấy password đăng nhập tại file `/var/jenkins_home/secrets/initialAdminPassword`.

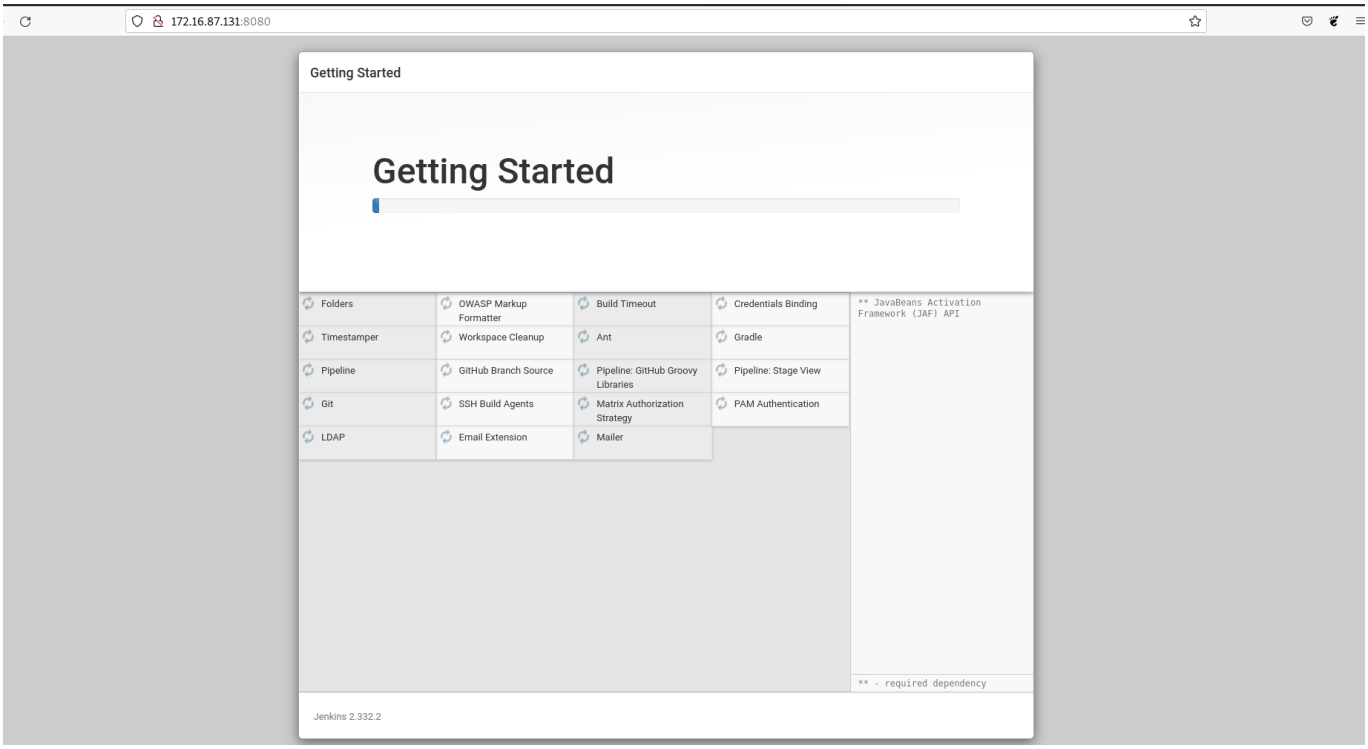
3. Hướng dẫn sử dụng

3.1. Cài đặt các plug-in

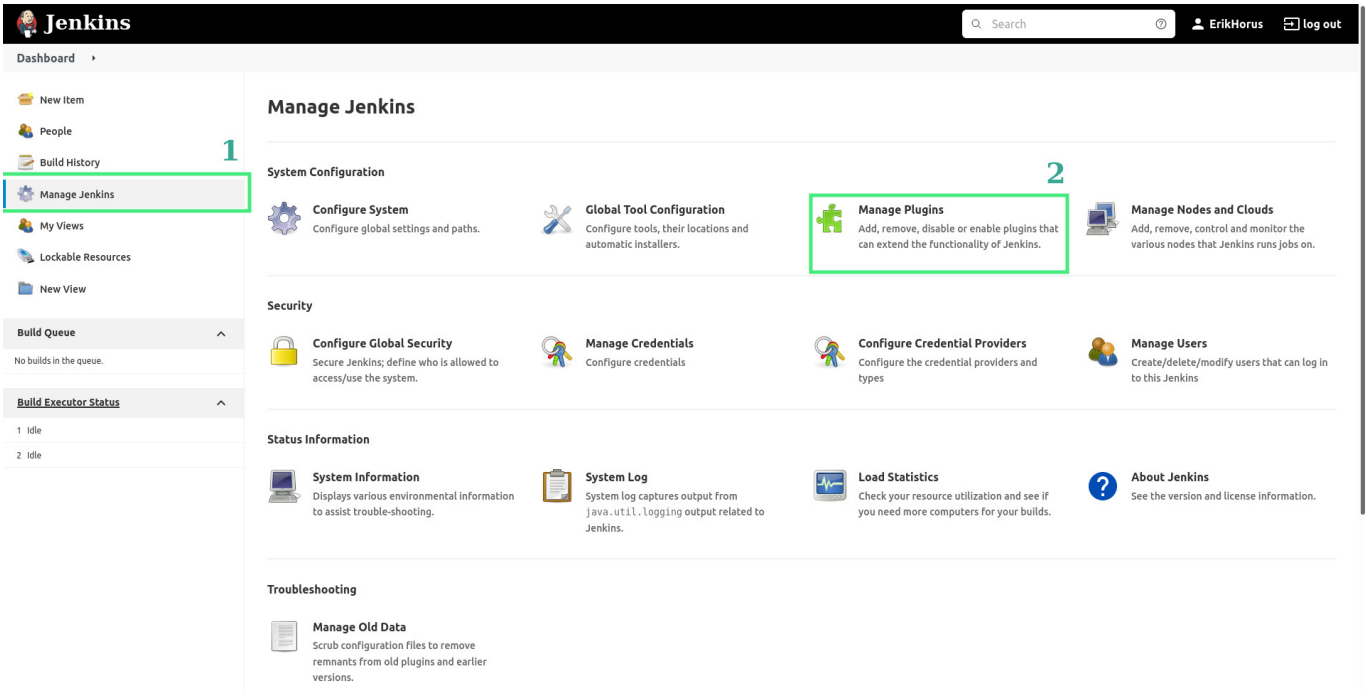
Bước 1: Nhập password đã lấy được vào Unlock Jenkins.



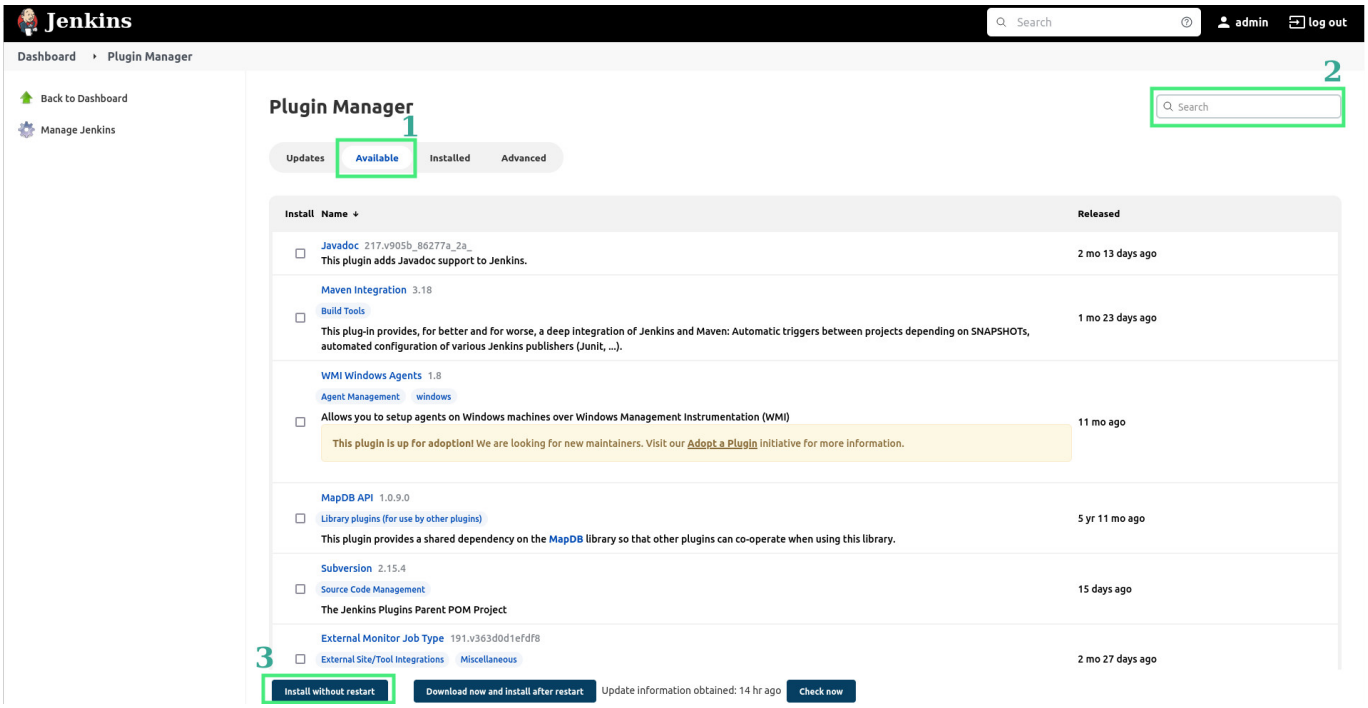
Bước 2: Chọn cài đặt các plug-in theo đề xuất của Jenkins.



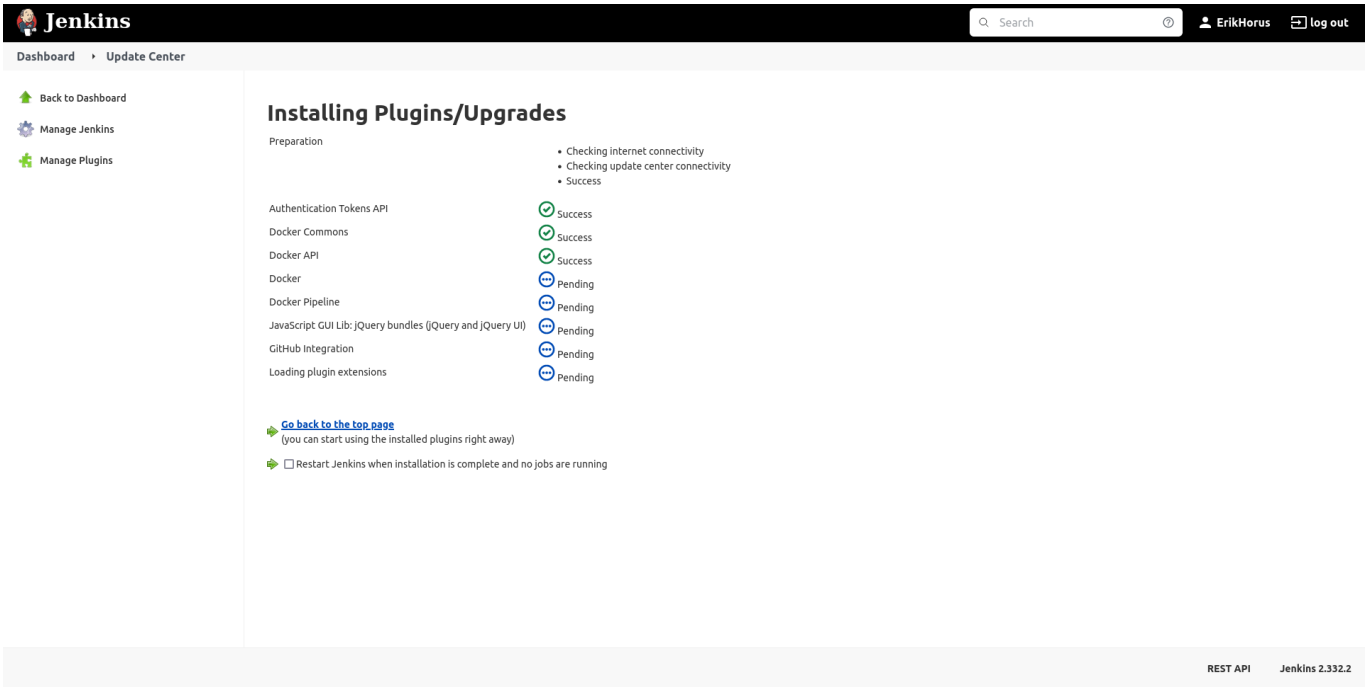
Bước 3: Cài đặt các Plug-in quan trọng.



- Tại giao diện dashboard của Jenkins chọn mục **Manage Jenkins** sau đó chọn tab **Manage plugins** -> chọn **Available** tag -> chọn **Search** và tìm với các keyword sau :
    - Docker plugin
    - Pipeline
    - github intergration
    - Github pull request builder
- Check vào các ô trước tên các plugin trên -> chọn **Install without restart**.

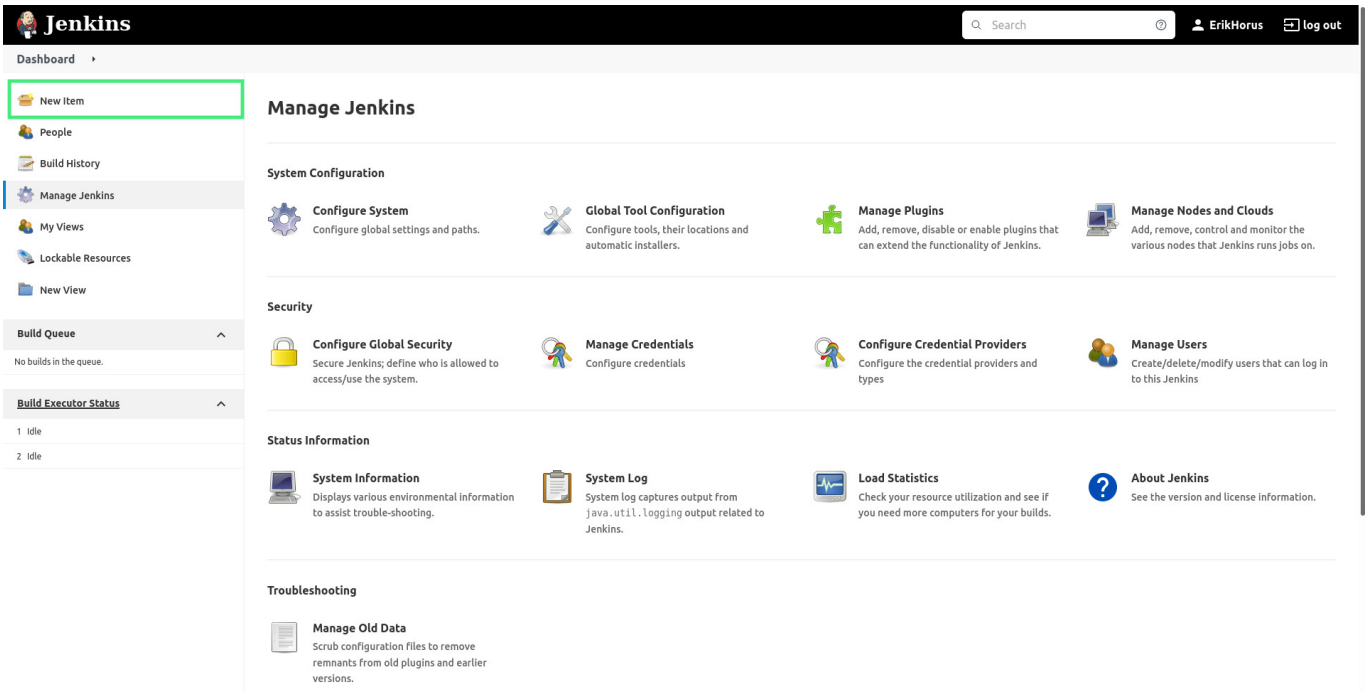


- Các plug-in đang được cài đặt.

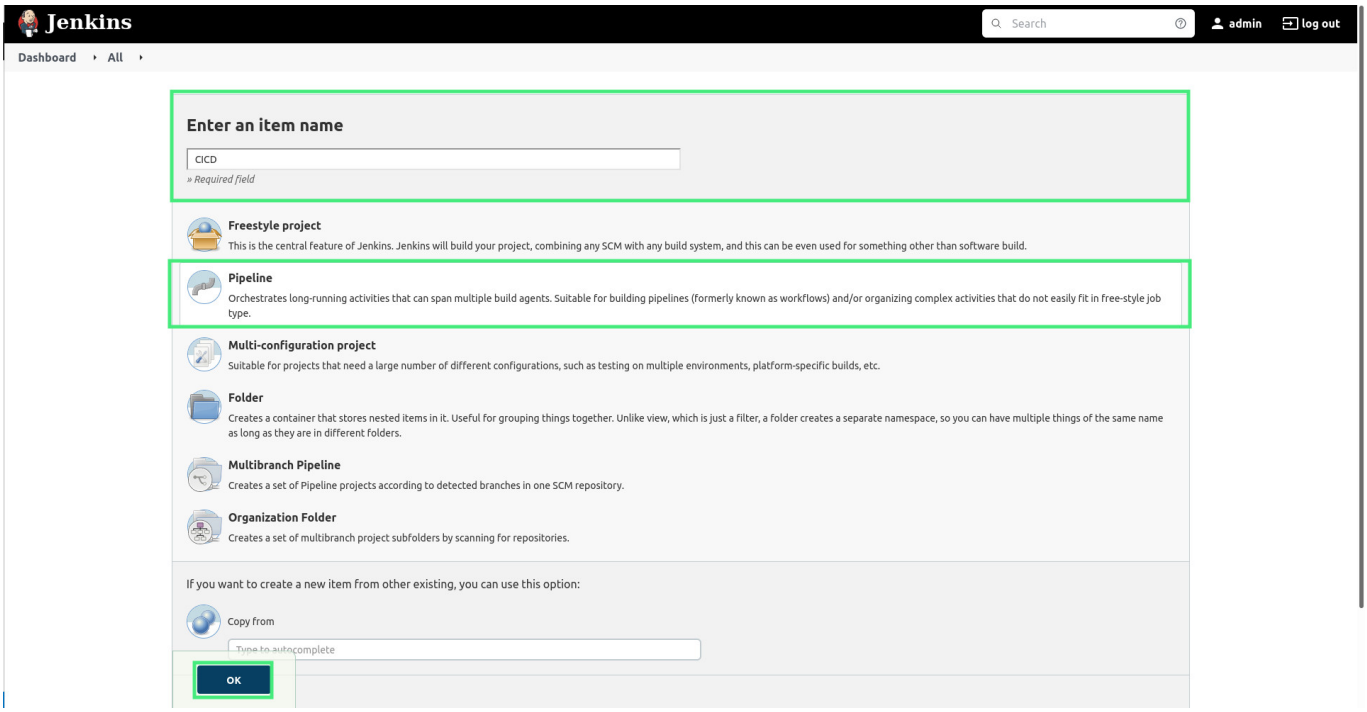


Bước 4: Tạo Job mới và cấu hình github.

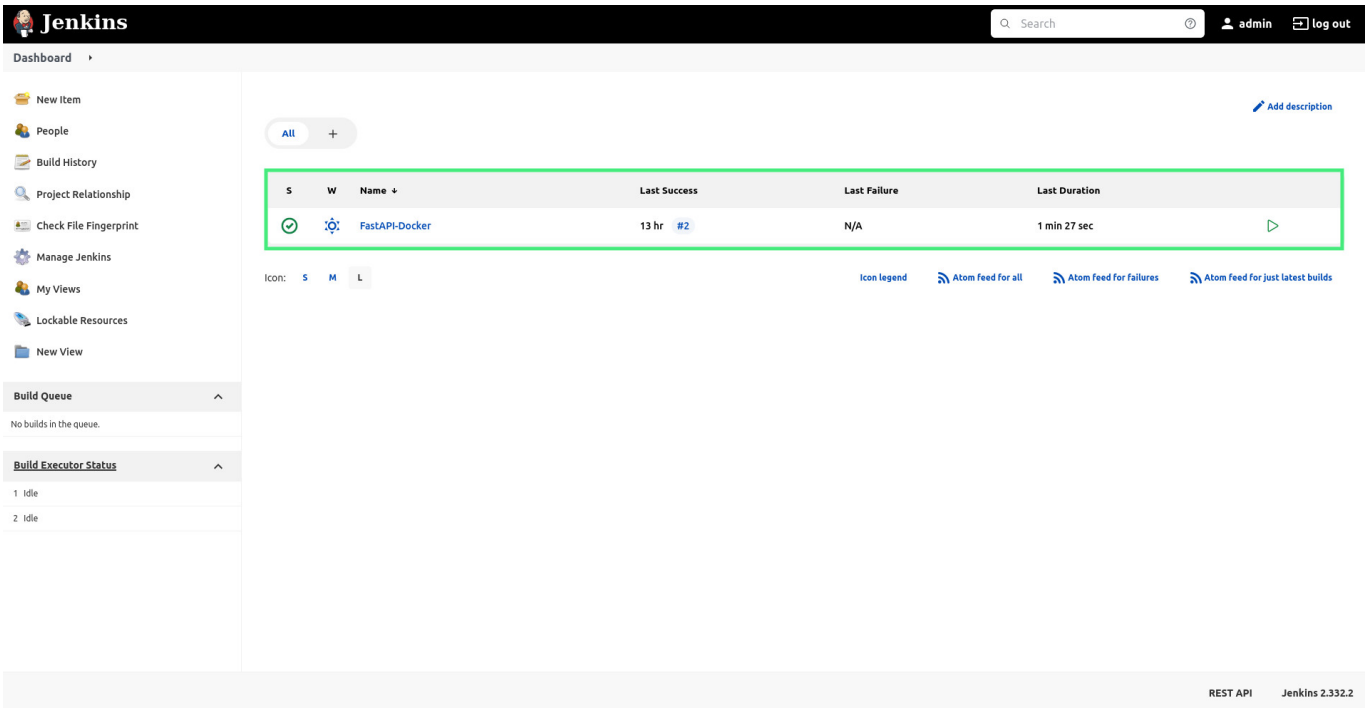
- Chọn **New item**.



- Đặt tên cho Job và chọn **Pipeline** -> Chọn **Ok**.

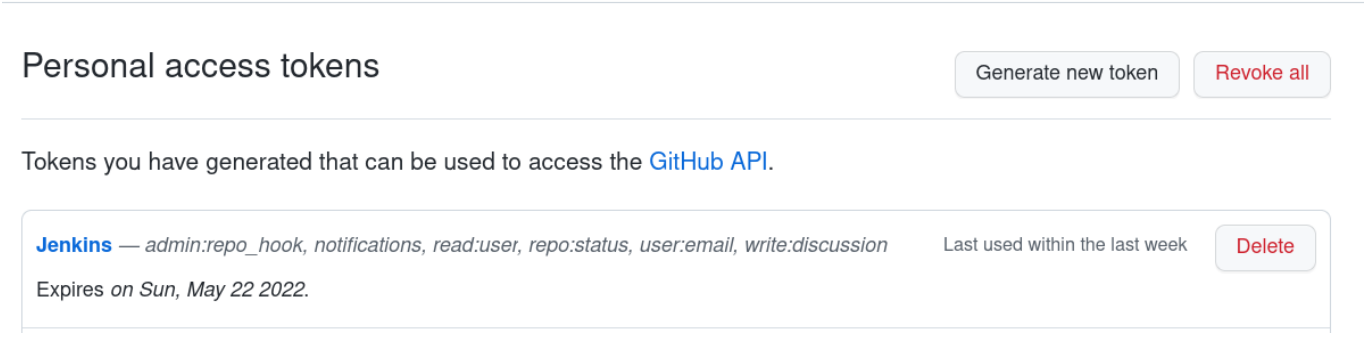


- Hoàn tất tạo Job.



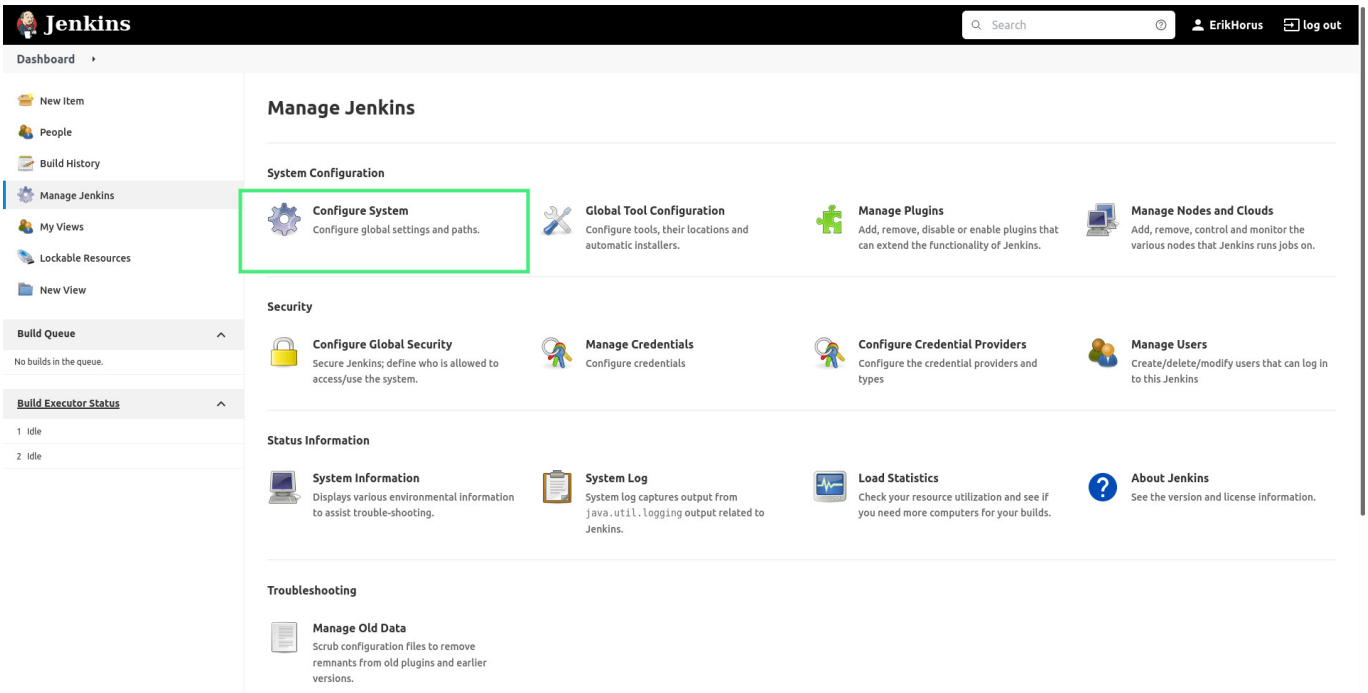
Bước 5: Tạo Github Personal Access Token cho Jenkins. Chú ý thêm đầy đủ các Role như trong hình. Sau đó copy Token để sử dụng.

- Tham khảo hướng dẫn tạo GPAT tại đây: **Github**.

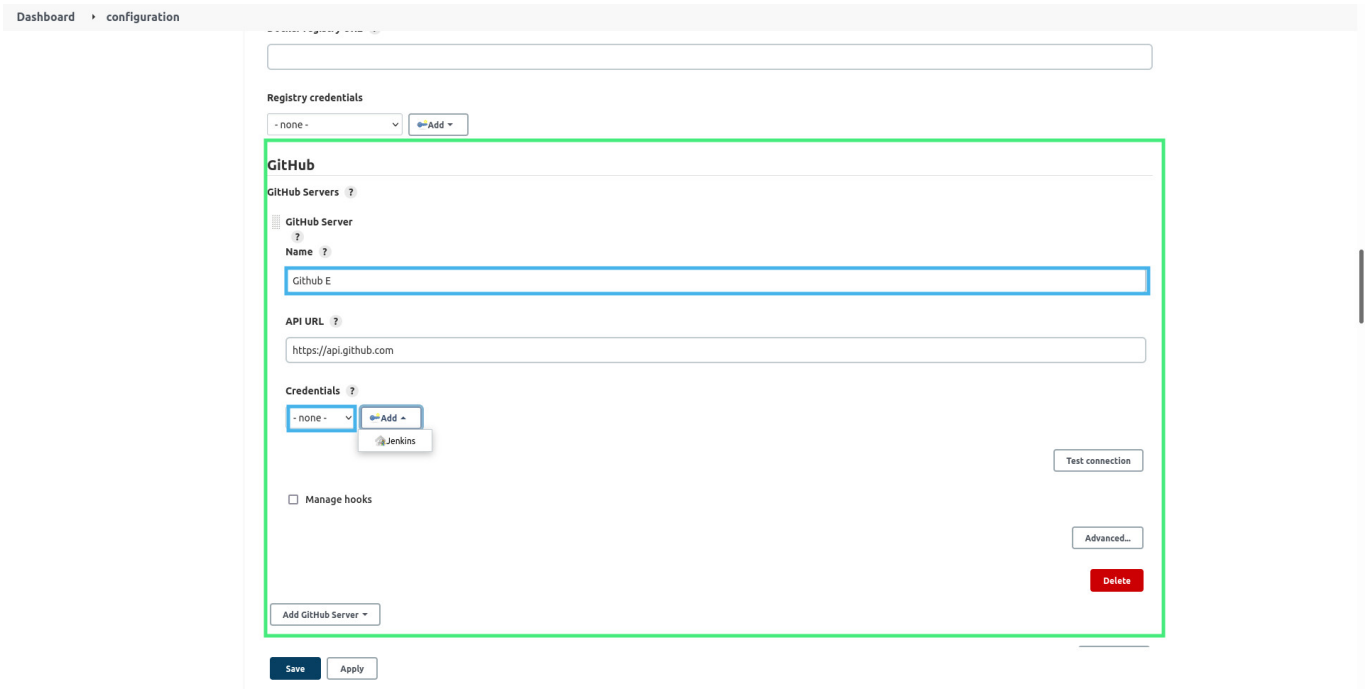


Bước 6: Cấu hình kết nối **Github**.

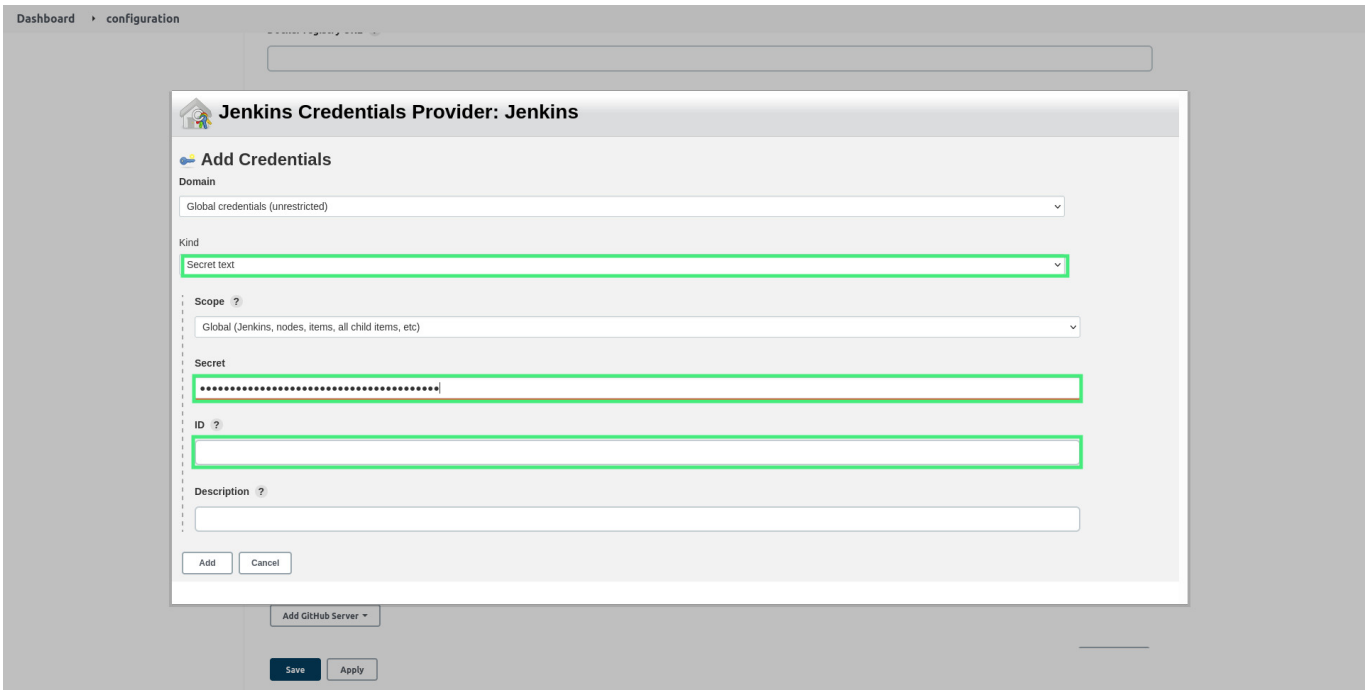
- Tại **Manage Jenkins** -> **Configure System**.



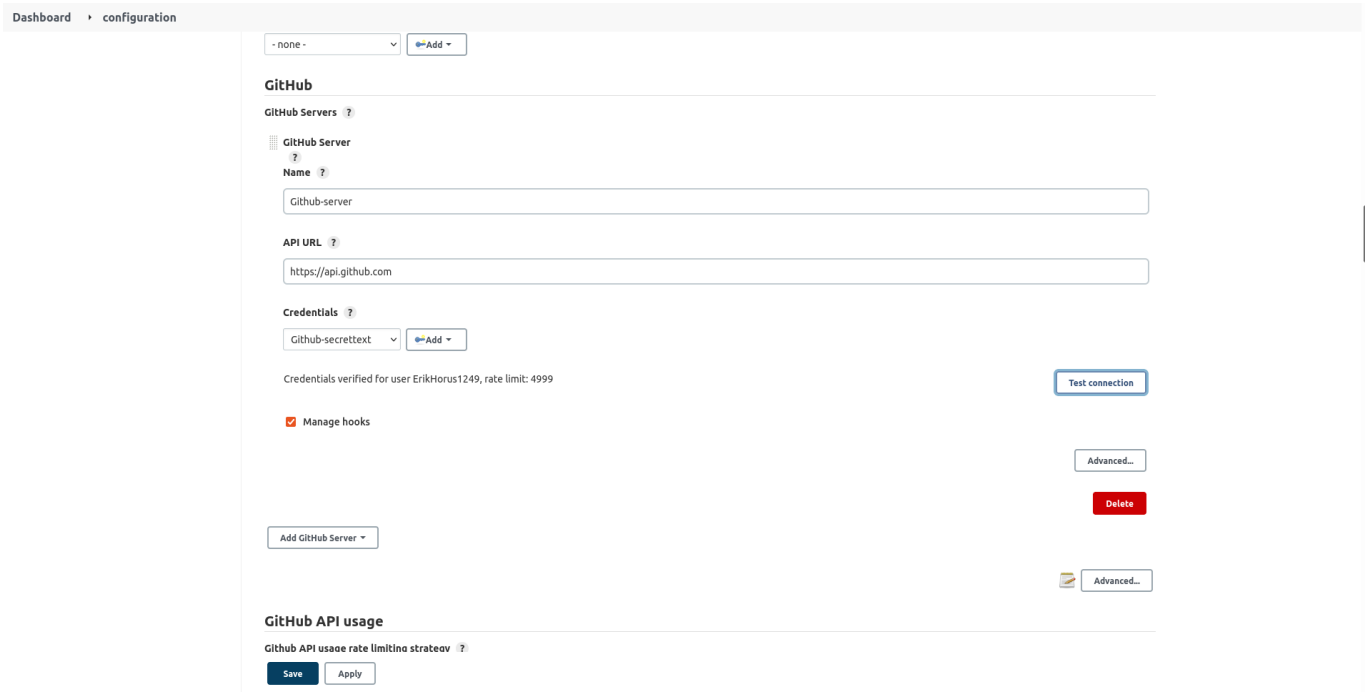
- Tại cấu hình Github thêm **Name** cho github server. VD: `Github-server`.
- Tại mục **Credentials** -> chọn **Add** -> chọn **Jenkins**.



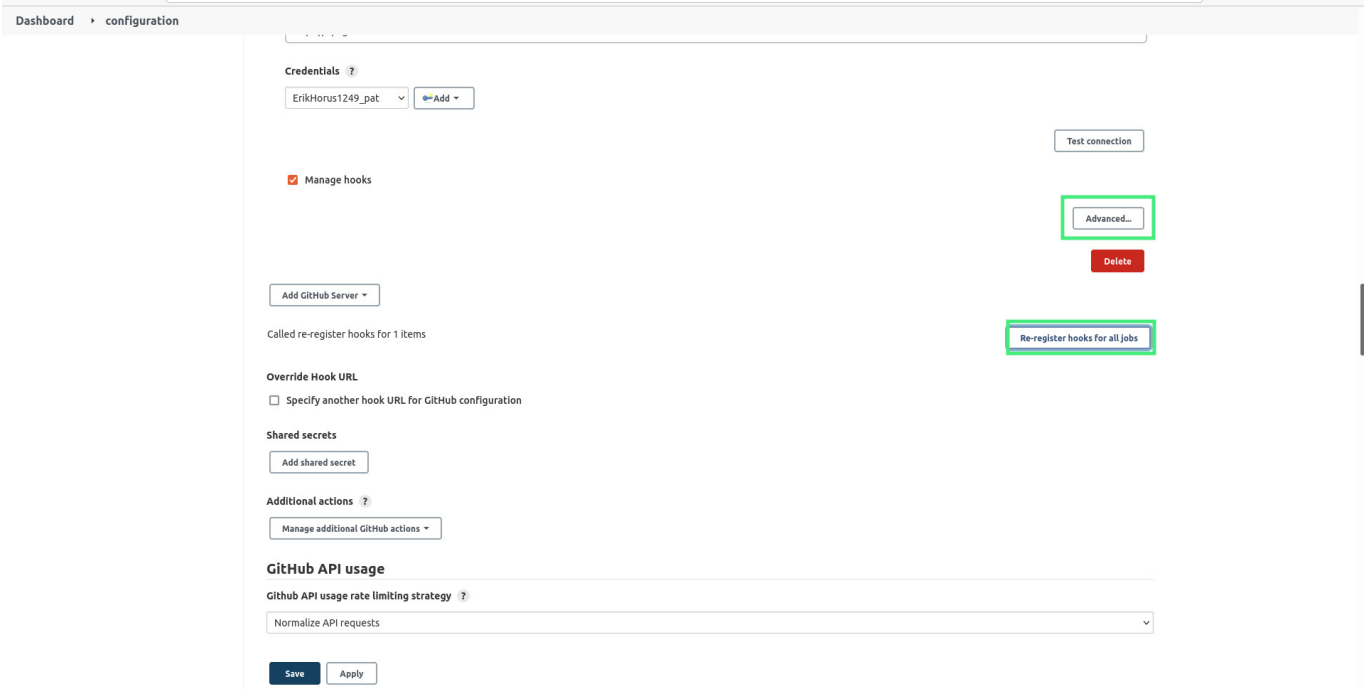
- Thêm các thông tin sau vào Form:
  - **Kind**: `Secret text`
  - **Sceret**: `paste Token tạo tại bước 5`
  - **ID**: `Github-secrettext`



- Sau khi tạo xong thì thêm vào mục bên trái và chọn **test connection** để kiểm tra.

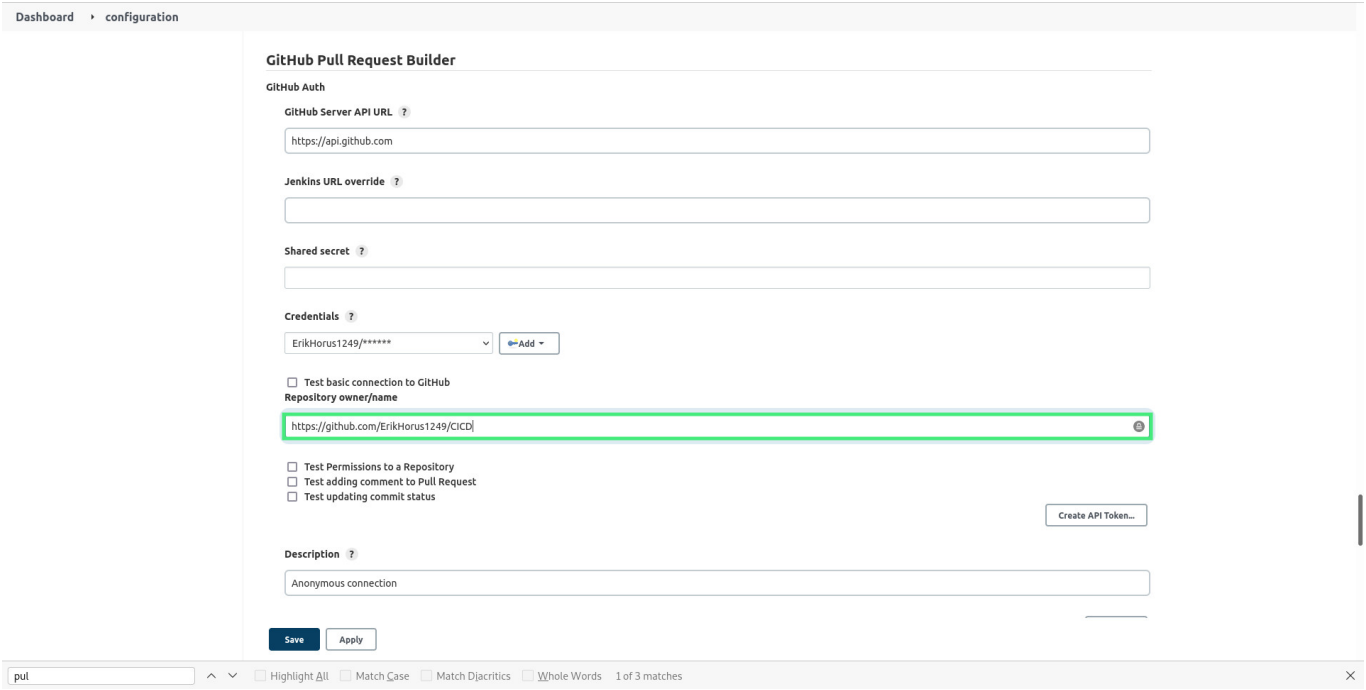


- Tiếp tục thêm Webhook cho Github phục vụ cho việc commit code thì Jenkins sẽ tự động build -> chọn **advanced** -> **Re-register hooks for all jobs**.

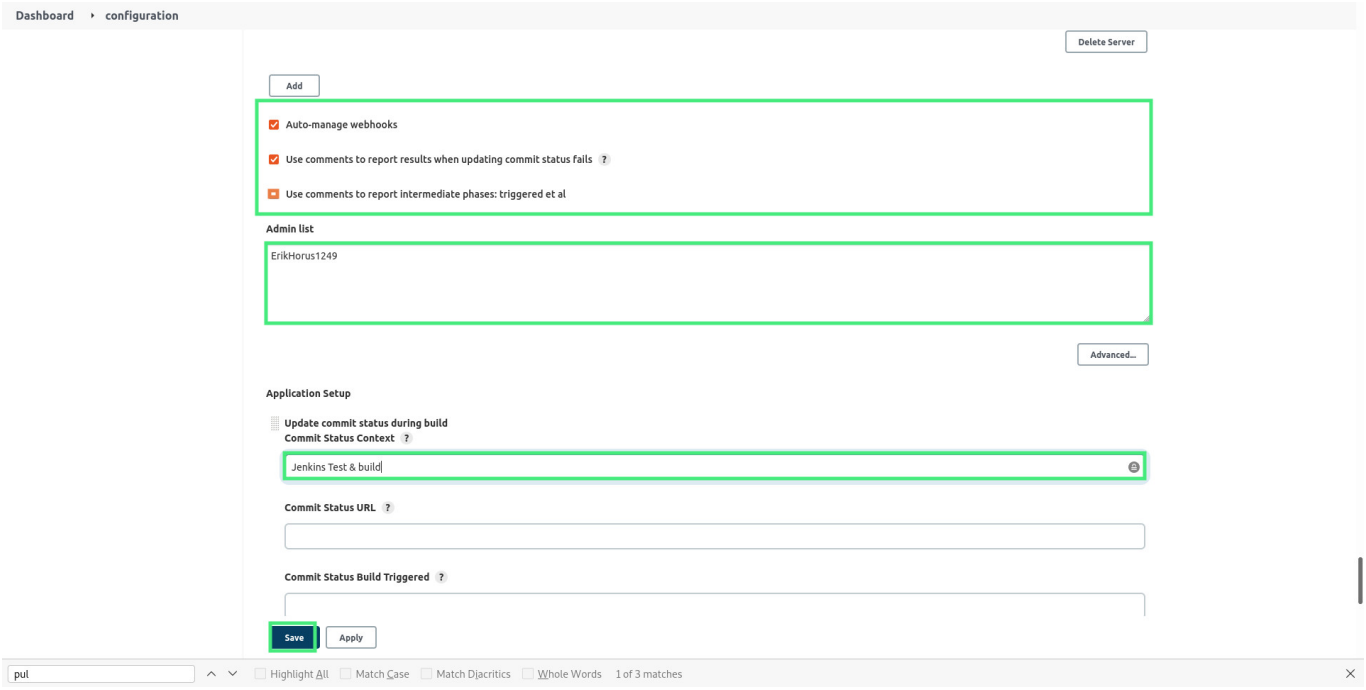


Bước 7: Cấu hình **Github pull request builder**.

- Tại Github pull request builder -> điền thông tin repo (chú ý chỉ điền github-user/name-of-repo).



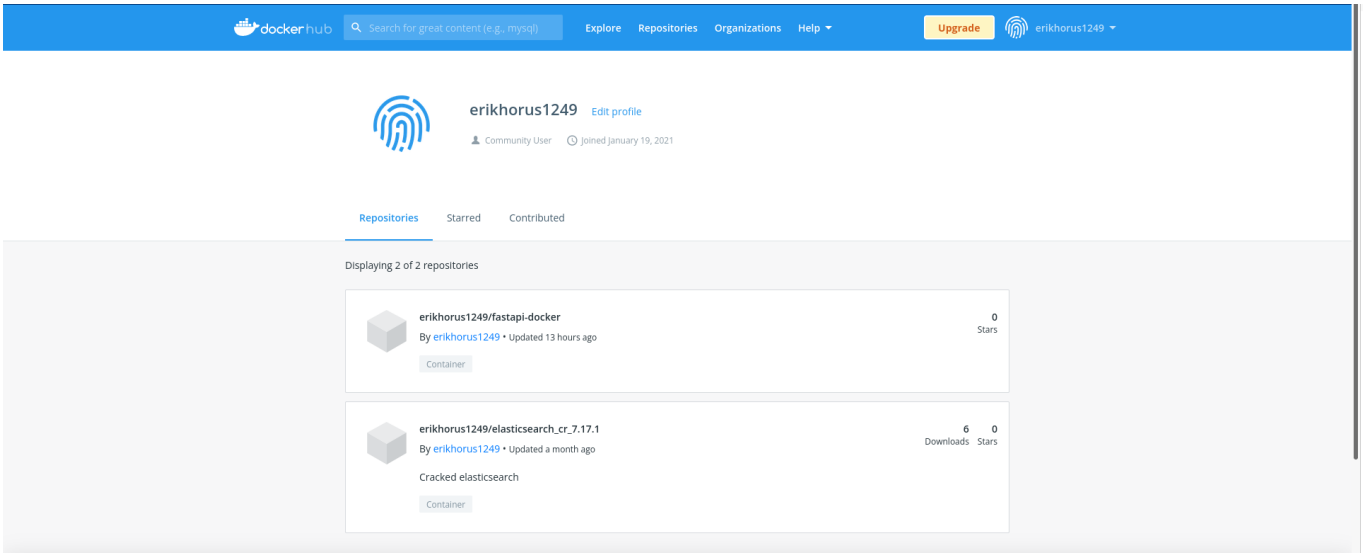
- Chọn Credentials như khi làm ở bước cấu hình github.



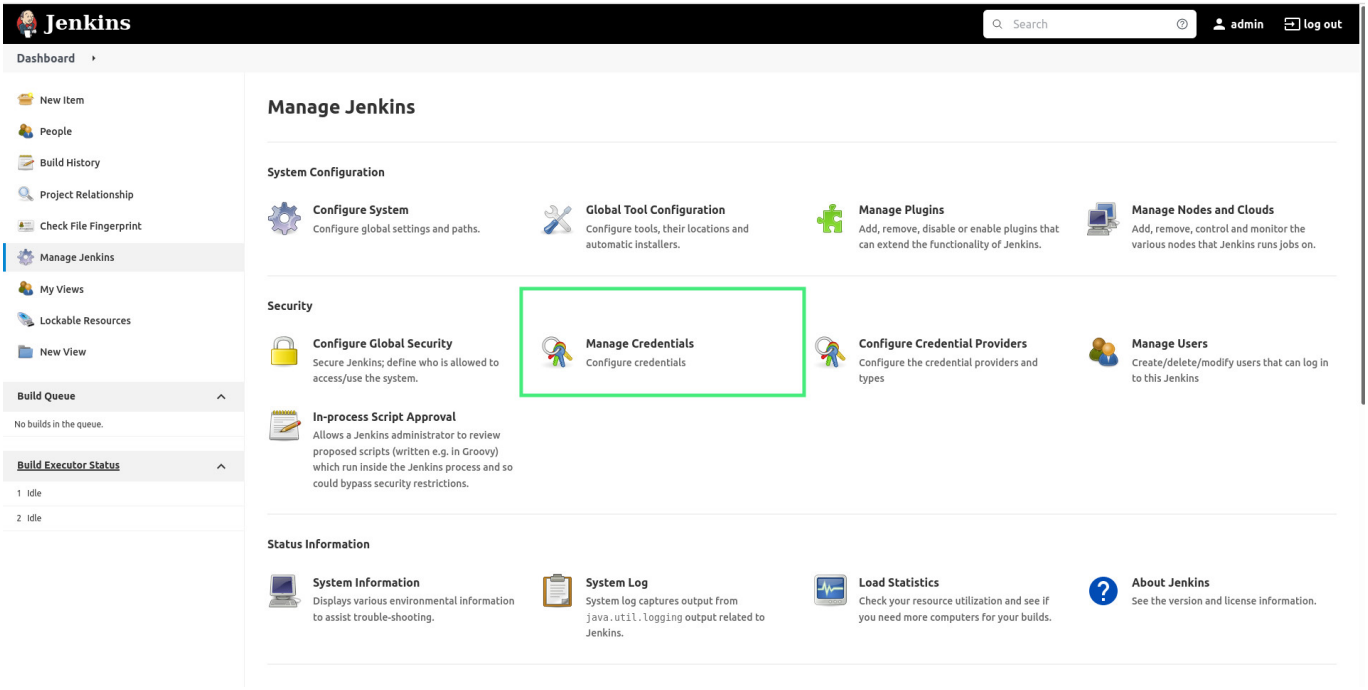
- Chọn 3 mục như hình -> thêm tên của admin (chính là github-account) -> thêm thông điệp trong context -> Chọn **Ok** để lưu.
- Sau khi cấu hình hoàn tất mỗi khi code mới được push lên Jenkins sẽ thực hiện qúa trình build phiên bản mới.

Bước 8: Tạo Personal Access Token cho **Dockerhub**.

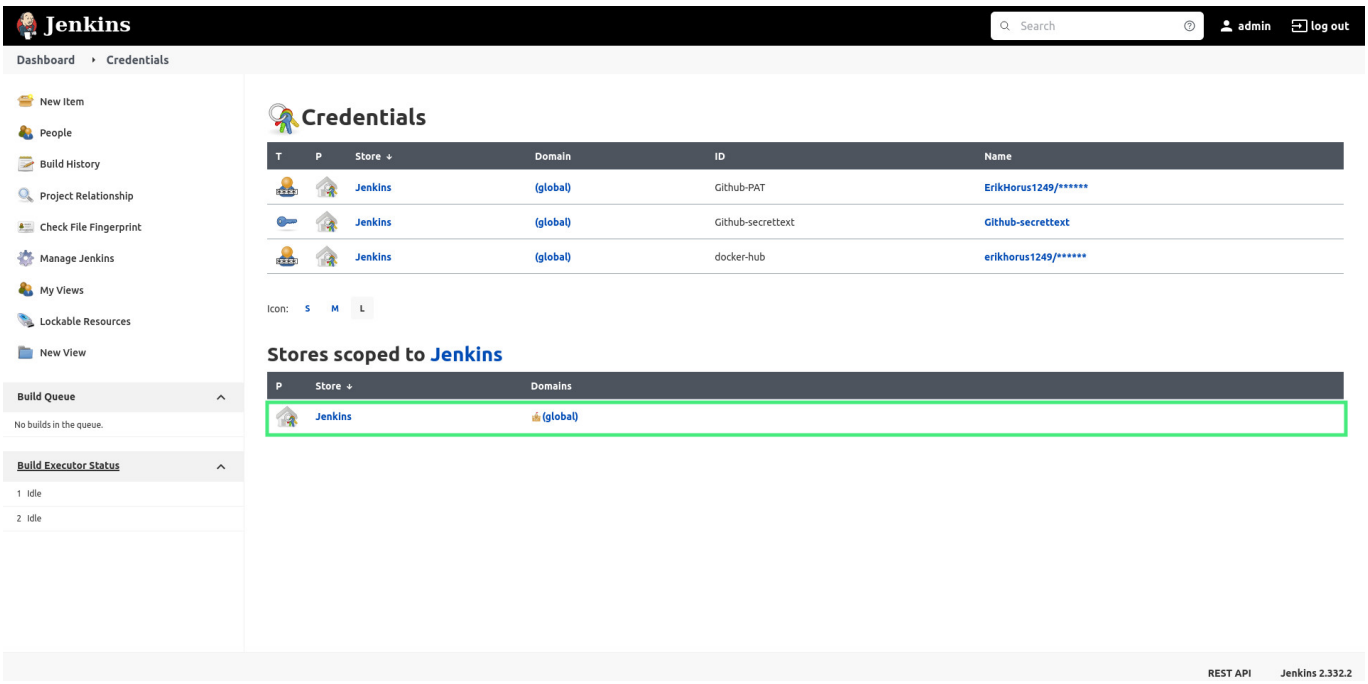
- Tạo tài khoản tại Hub.Docker.com nếu chưa có tài khoản.
- Lưu lại thông tin username/password sử dụng khi tạo Access Token.



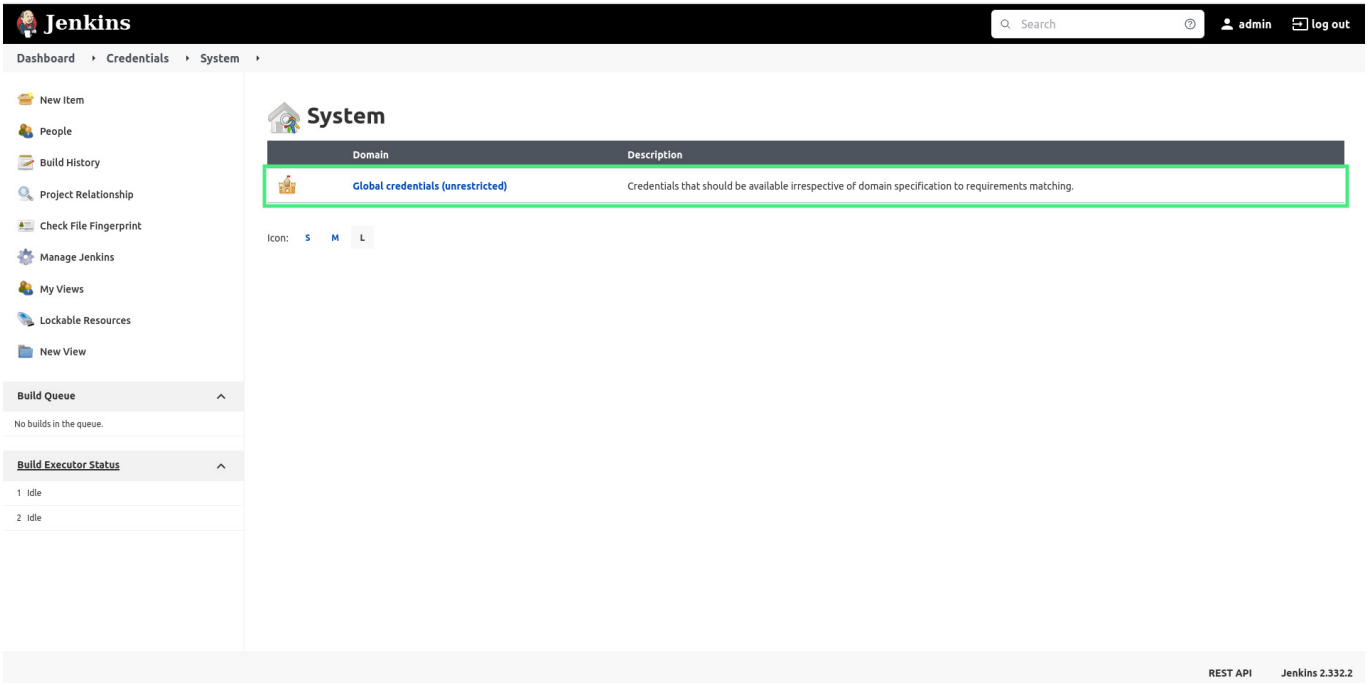
- Chọn **Manage Jenkins** -> **Manage Credentials**.



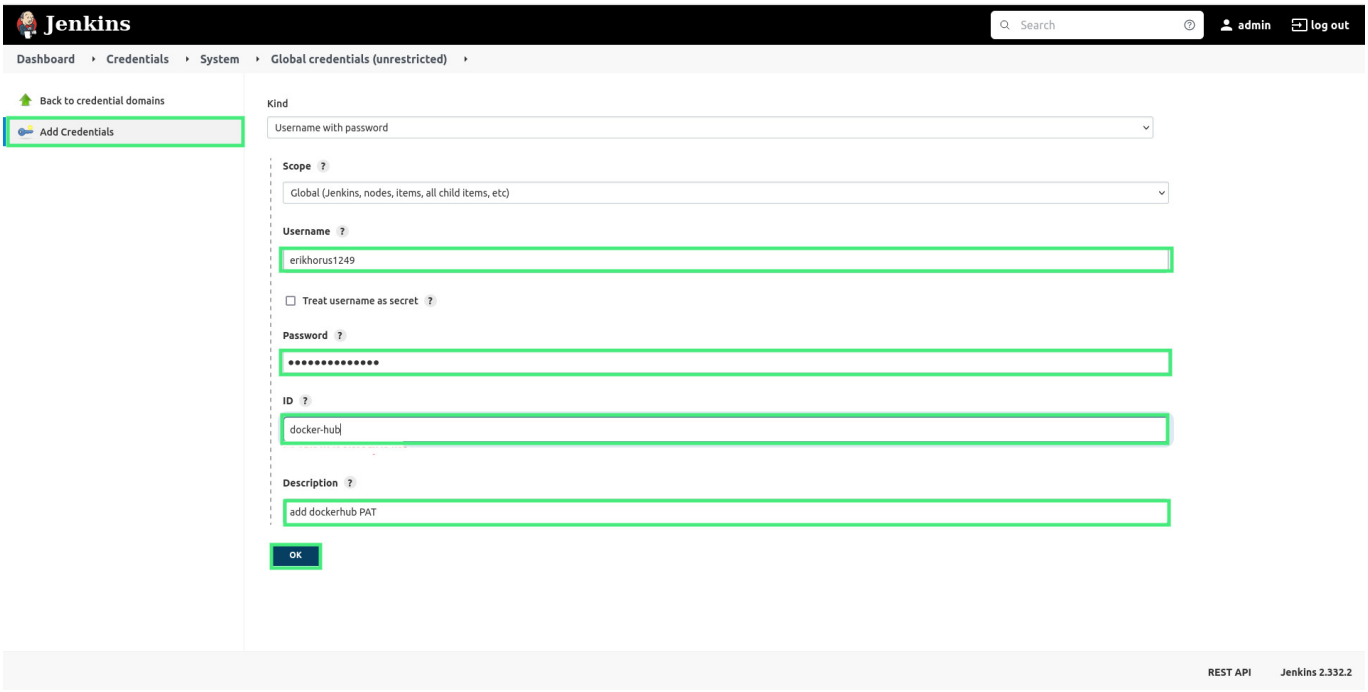
- Chọn như hình sau.



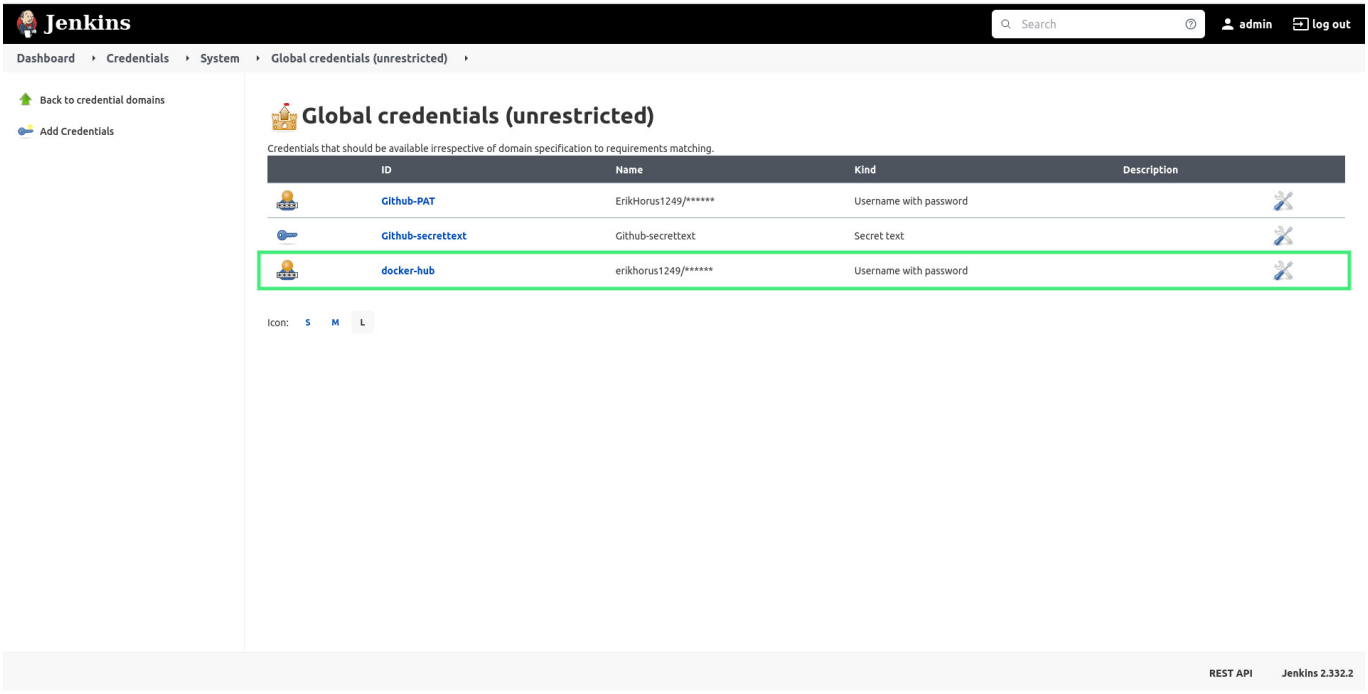
- Chọn **Global credentials**.



- Chọn **Add credentials**
- Điền các thông tin cần thiết theo mẫu sau:
  - **Username**: `<dockerhub-username>`
  - **Password**: `<dockerhub-password>`
  - **ID**: docker-hub
  - **Descriptions**: `Tùy chọn khi nhập`
- Sau khi điền chọn **Ok** để hoàn tất.



- Sau khi tạo **Credential**.



3.2. Chuẩn bị source code dự án

3.2.1. Lưu ý về source code

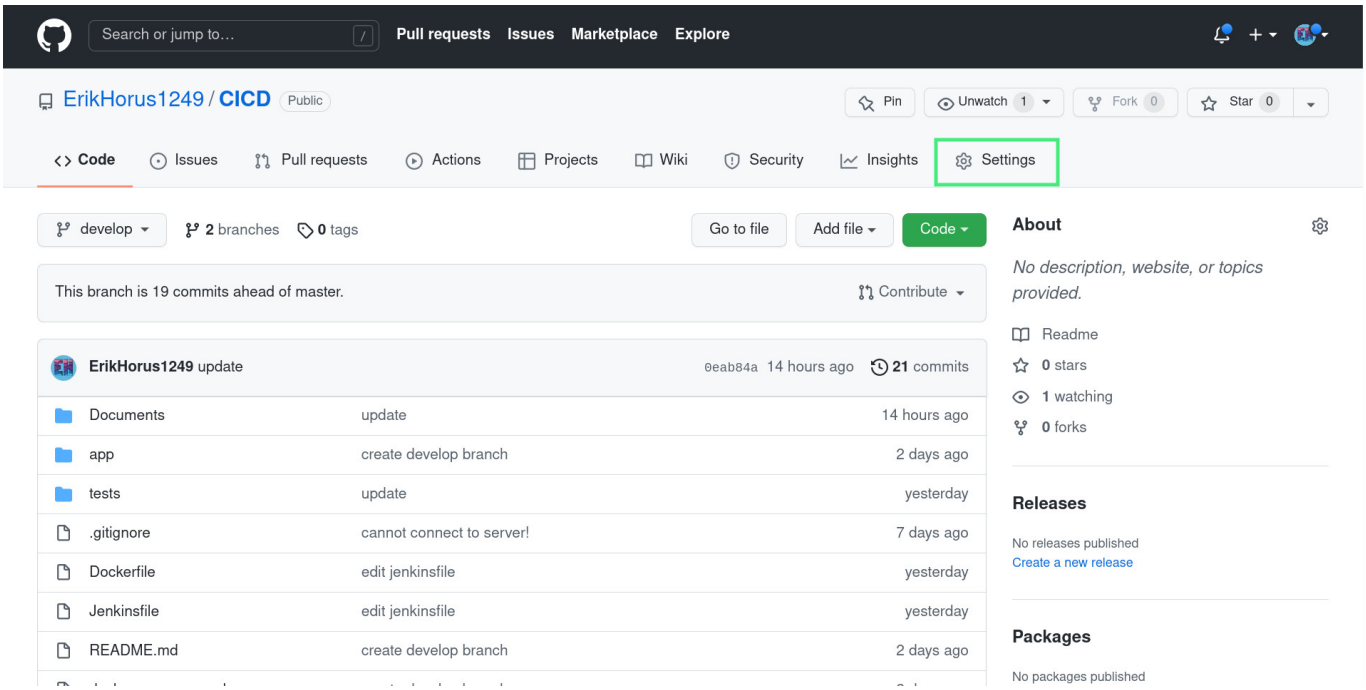
- Dự án sử dụng trong demo là một project FastAPI triển khai một số API đã có Dockerfile và docker-compose.yml.
- Dự án sử dụng poetry để quản lý dependency và package. Chi tiết sử dụng poetry tham khảo tại : **Python-poetry**.
- Quan trọng nhất là file pipeline : **Jenkinsfile** chưa thông tin test và build của dự án. Tham khảo chi tiết về Jenkinsfile tại: **Github/ErikHorus1249**.

```
📦app
 ┣ 📜__init__.py
 ┗ 📜main.py
📦tests
 ┣ 📜__init__.py
 ┗ 📜test_server.py
 ┣ 📜.gitignore
 ┣ 📜Dockerfile
 ┣ 📜Jenkinsfile
 ┣ 📜README.md
 ┣ 📜docker-compose.yml
 ┣ 📜poetry.lock
 ┗ 📜pyproject.toml
```

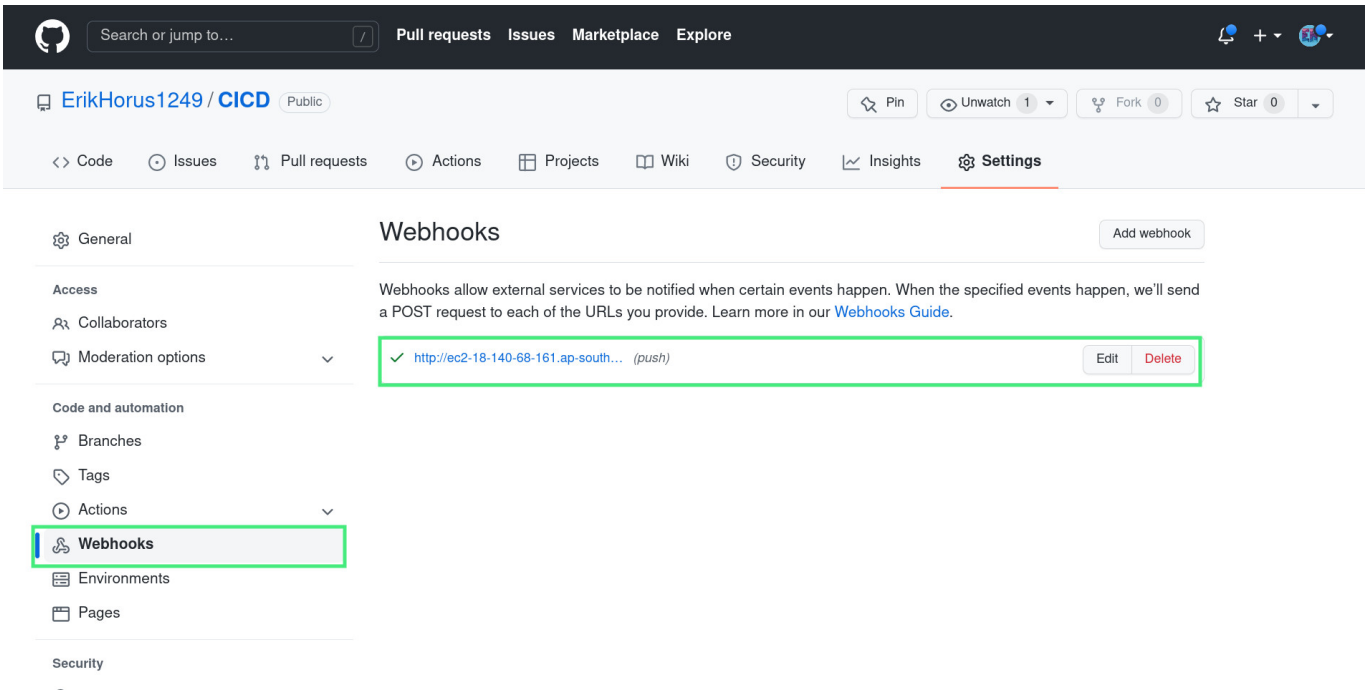
- Đường dẫn tham khảo source code dự án: **Github**

3.2.1. Kiểm tra wwebhook đã được thêm vào Github repository chưa.

- Tại chọn **Settings**.

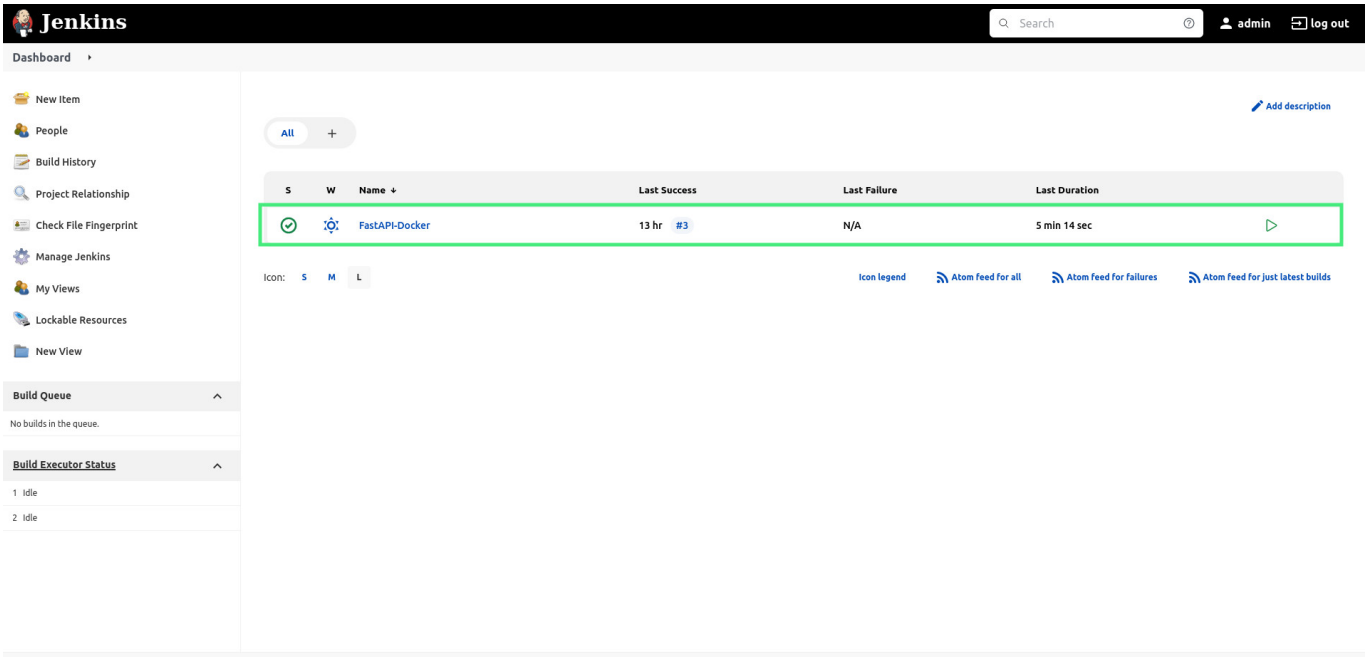


- Chọn **Webhooks** và kiểm tra xem đã Webhook được thêm vào chưa.
- Nếu chưa có thực hiện lại phần cấu Gidhub **Bước 6**.

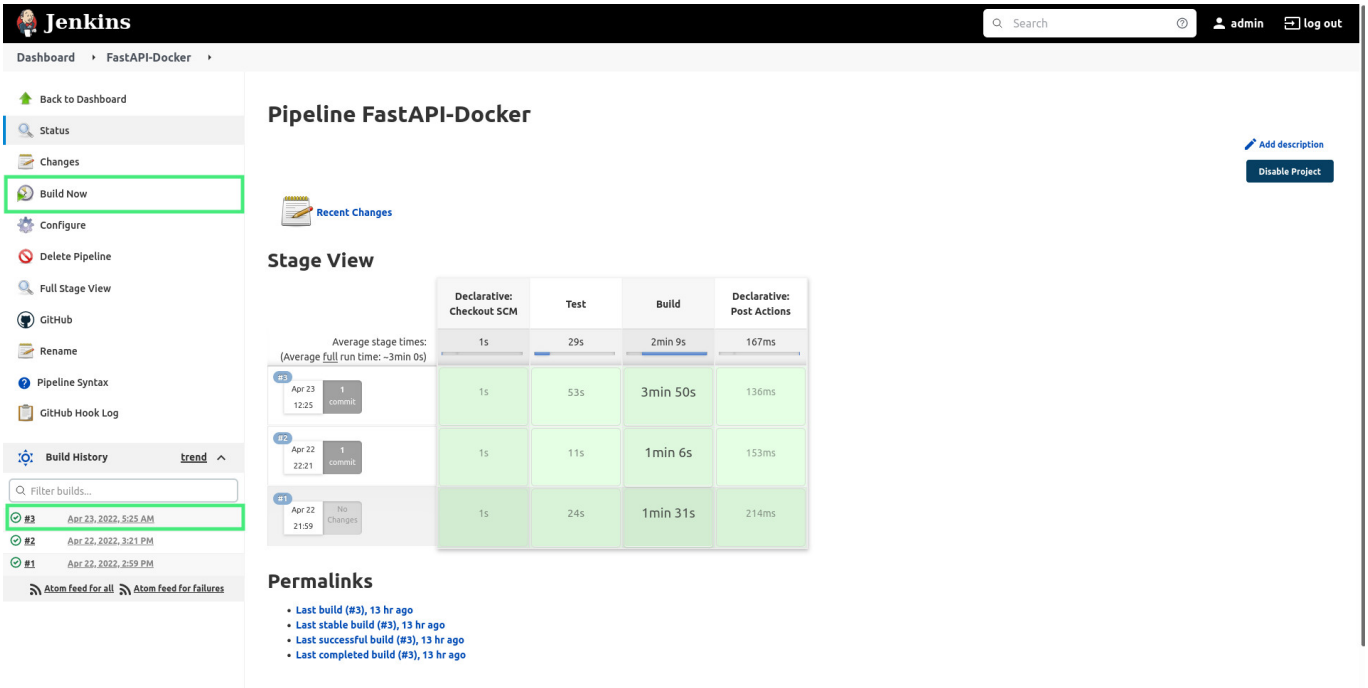


3.3. Build dự án

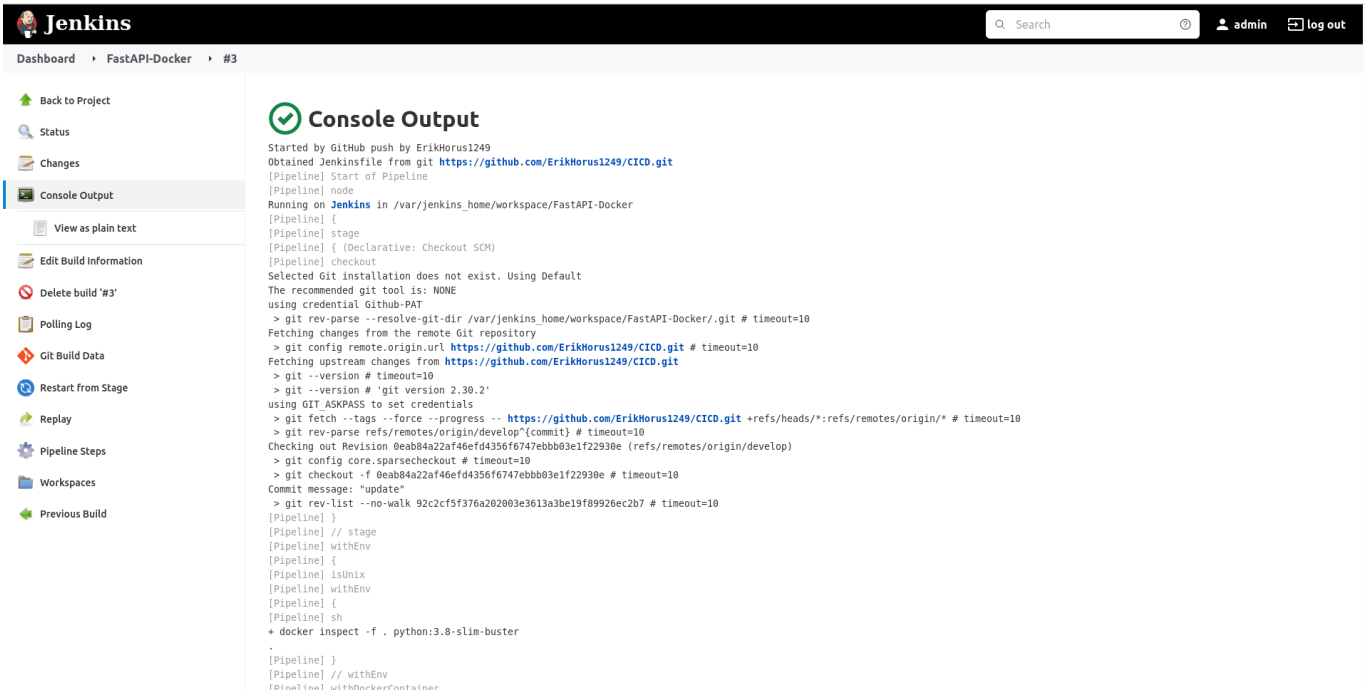
Bước 1: Chọn vào tên của Job -> chọn **Build now**.



- Chọn vào mục Build History để xem thông tin build chi tiết qua console.



- Quan sát và kiểm tra khi có lỗi phát sinh trong quá trình test và build.



- Quá trình build sẽ diễn ra tự động khi có code mới được push lên.

[*]Chúc ace thường doc vui vẻ. Mọi ý kiến góp ý, chỉnh sửa xin để lại tại phần Issues . Chân thành cảm ơn !

Nguồn tham khảo:

- poetry
- Topdev
- File-tree-extension
- Imgur
- Stackedit