

## Assignment 3

Môn học: **Lập trình an toàn & Khai thác lỗ hổng phần mềm**

Tên chủ đề: **Hiện thực pipeline CI/CD với Jenkins**

Mã môn học: NT521 – Năm học 2023-2024

### 1. YÊU CẦU CHUNG

Sinh viên thực hiện yêu cầu cài đặt và cấu hình Jenkins kết hợp với 1 dự án phần mềm trên kho lưu trữ GitHub để hiện thực 1 pipeline CI/CD đơn giản.

#### a) Chuẩn bị môi trường

- 1 máy ảo Linux có cài đặt **Docker**
- 1 tài khoản GitHub
- Source của dự án phần mềm: **sample-app.zip**

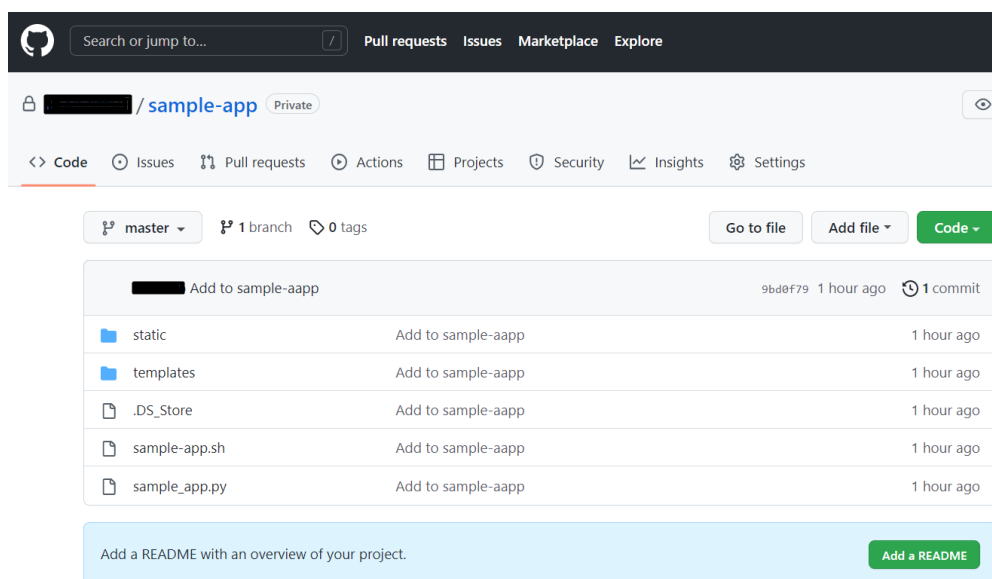
#### b) Các yêu cầu cần thực hiện

Sinh viên thực hiện theo các bước hướng dẫn bên dưới, chụp hình báo cáo minh chứng kết quả của từng bước.

### 2. NỘI DUNG THỰC HIỆN

#### a) Commit Sample App lên GitHub

Sinh viên giải nén **sample-app.zip**, sau đó tạo GitHub repository để commit sample-app, với các bước tương tự ở **Bài TH 1**. Sau khi thực hiện thành công, ta thu được kết quả như bên dưới.



**b) Sửa đổi Sample App và push thay đổi lên Git**

Do mặc định Sample App và Jenkins Docker cùng dùng port 8080, nên ta thay đổi port trong mã nguồn của Sample App.

**Bước 1.** Mở tập tin **sample\_app.py**, chỉnh sửa port 8080 thành 5050 ở dòng bên dưới:

```
if __name__ == "__main__":  
    sample.run(host="0.0.0.0", port=5050)
```

**Bước 2.** Mở tập tin **sample-app.sh**, cũng chỉnh sửa port 8080 thành 5050:

```
...  
echo "COPY sample_app.py /home/myapp/" >> tempdir/Dockerfile  
echo "EXPOSE 5050" >> tempdir/Dockerfile  
echo "CMD python /home/myapp/sample_app.py" >> tempdir/Dockerfile  
  
cd tempdir  
docker build -t sampleapp .  
docker run -t -d -p 5050:5050 --name samplerunning sampleapp  
docker ps -a
```

**Bước 3.** Build và kiểm tra sample-app

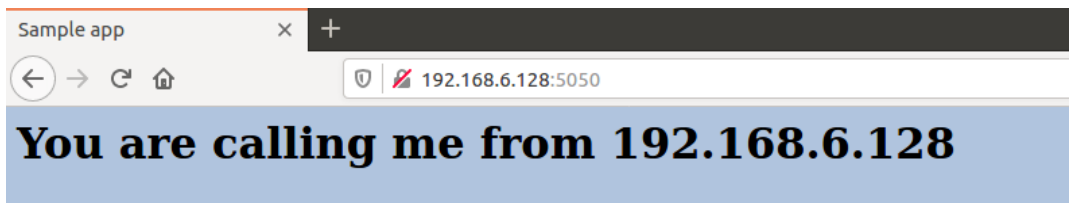
Thực thi file **sample-app.sh** để chạy ứng dụng với port mới 5050

```
$ sudo bash ./sample-app.sh
```

Mở 1 terminal khác hoặc trình duyệt, kiểm tra đường dẫn <http://localhost:5050> hoặc [http://<ip\\_linux\\_vm>:5050](http://<ip_linux_vm>:5050)

```
$ curl localhost:5050
```

Hoặc



**Bước 4.** Push sự thay đổi lên GitHub

Sinh viên thực hiện commit và push code với thông điệp **“Changed port from 8080 to 5050”**, trình bày step-by-step có hình ảnh minh chứng và hình ảnh kết quả trên Github.

**c) Tải và thiết lập chạy Jenkins Docker Image**

**Bước 1.** Tải Jenkins Docker image

Gõ lệnh để tải xuống Jenkins:

```
$ sudo docker pull jenkins/jenkins
```

**Bước 2.** Khởi chạy Jenkins Docker container

Lệnh sau sẽ khởi chạy Jenkins Docker container và sau đó cho phép các lệnh Docker được thực thi bên trong máy chủ Jenkins.

```
$ sudo docker run --rm -u root -p 8080:8080 -v jenkins-
data:/var/jenkins_home -v $(which docker):/usr/bin/docker -v
/var/run/docker.sock:/var/run/docker.sock -v "$HOME":/home --name
jenkins_server jenkins/jenkins:lts
```

**Bước 3.** Máy chủ Jenkins đã khởi chạy xong khi có dòng thông báo như bên dưới.

```
Jenkins initial setup is required. An admin user has been created and a
password generated.
Please use the following password to proceed to installation:

399b1d4a2c22443aa33a2bb17d669408

This may also be found at: /var/jenkins_home/secrets/initialAdminPasswor
d

*****
*****
*****

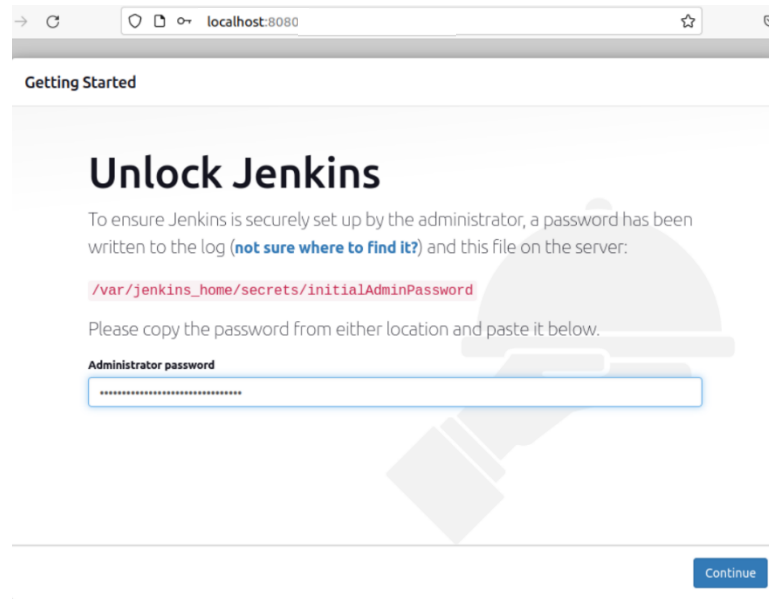
2022-09-11 16:57:13.300+0000 [id=28] INFO jenkins.InitReactorRunne
r$1#onAttained: Completed initialization
2022-09-11 16:57:13.383+0000 [id=22] INFO hudson.lifecycle.Lifecyc
le#onReady: Jenkins is fully up and running
```

Lưu ý bên trên có dòng password **cần lưu lại** để sử dụng về sau. Sau khi khởi chạy thành công, có thể hình dung hệ thống đang chạy theo biểu đồ dưới đây, để có cái nhìn về các mức độ hệ thống Docker-inside-Docker.

```
+-----+
|Your Computer's Operating System|
| +-----+ |
| |Linux VM| | | | |
| | +-----+ |
| | | Docker container | |
| | | +-----+ |
| | | | Jenkins server | |
| | | | +-----+ |
| | | | | Docker container | |
| | | | | +-----+ |
| | | | +-----+ |
| | | +-----+ |
| | +-----+ |
| +-----+ |
+-----+
```

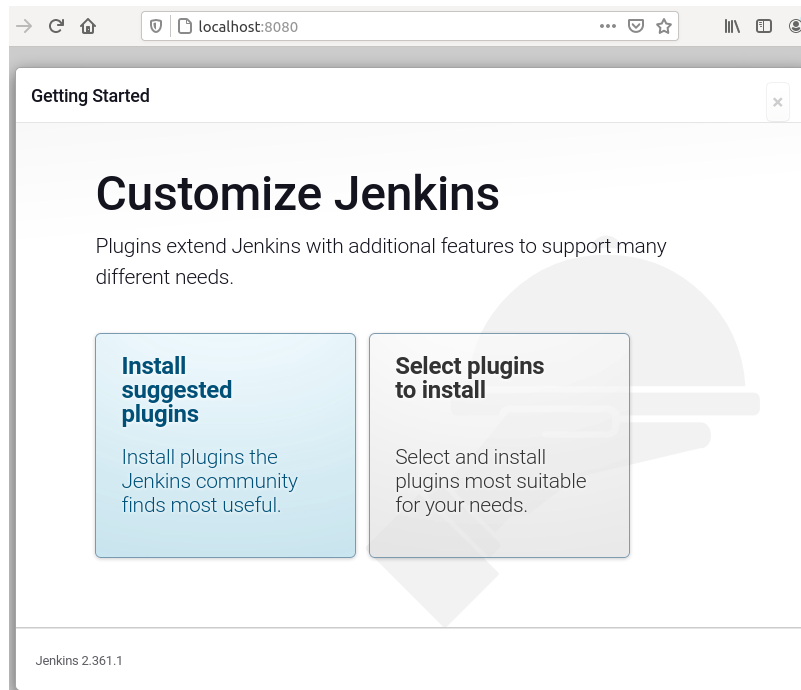
#### d) Cấu hình Jenkins

**Bước 1.** Mở trình duyệt và truy cập <http://localhost:8080/> hoặc [http://<ip\\_linux\\_vm>:8080/](http://<ip_linux_vm>:8080/) và nhập mật khẩu admin đã lưu lại ở Bước trước.

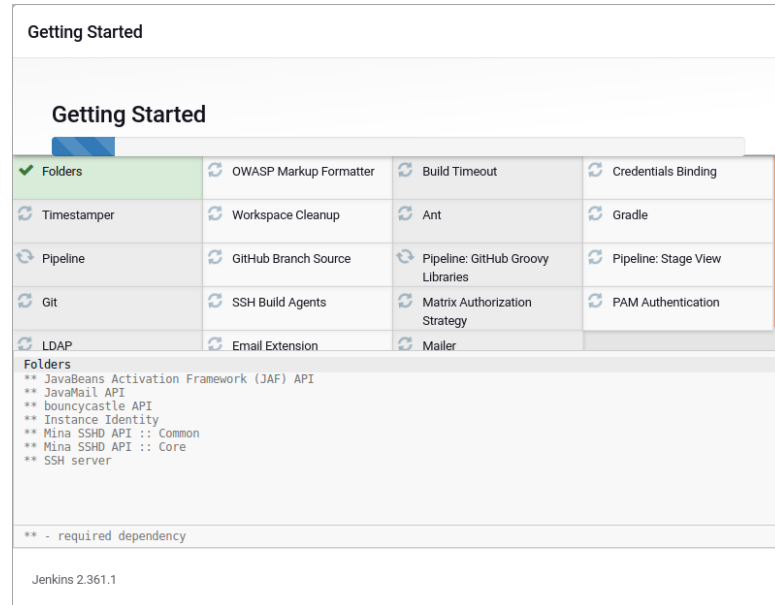


**Bước 2.** Cài đặt các Jenkins plugins khuyến nghị

Chọn vào Cài đặt các plugin được đề xuất và đợi Jenkins tải xuống và cài đặt các plugin.



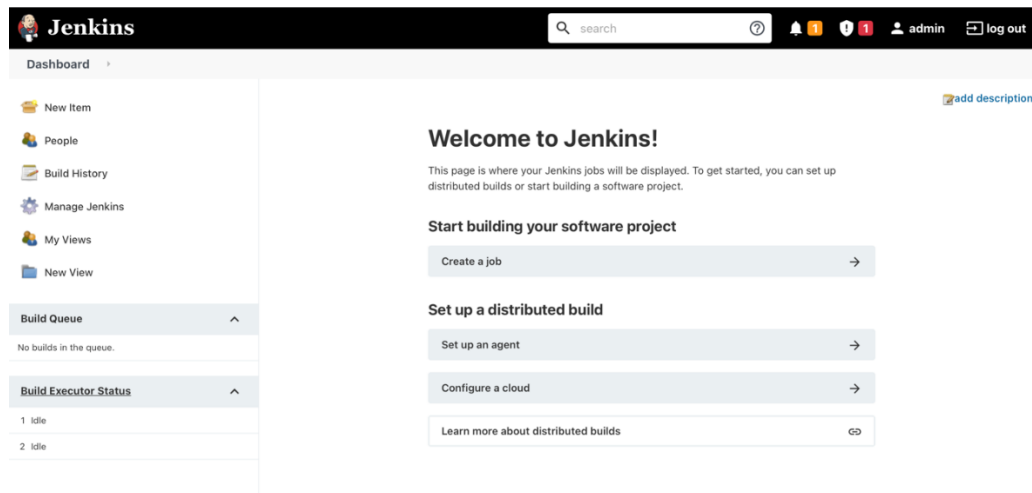
Trong cửa sổ terminal, ta sẽ thấy các thông báo log khi quá trình cài đặt được tiến hành. Giai đoạn này sẽ **tốn khá nhiều thời gian**.



**Bước 3.** Bỏ qua tạo tài khoản admin bằng cách chọn *Skip and continue as admin*.

**Bước 4.** Trong cửa sổ *Instance Configuration*, chọn tiếp *Save and Finish*.

**Bước 5.** Bắt đầu sử dụng Jenkins bằng cách chọn *Start using Jenkins*.



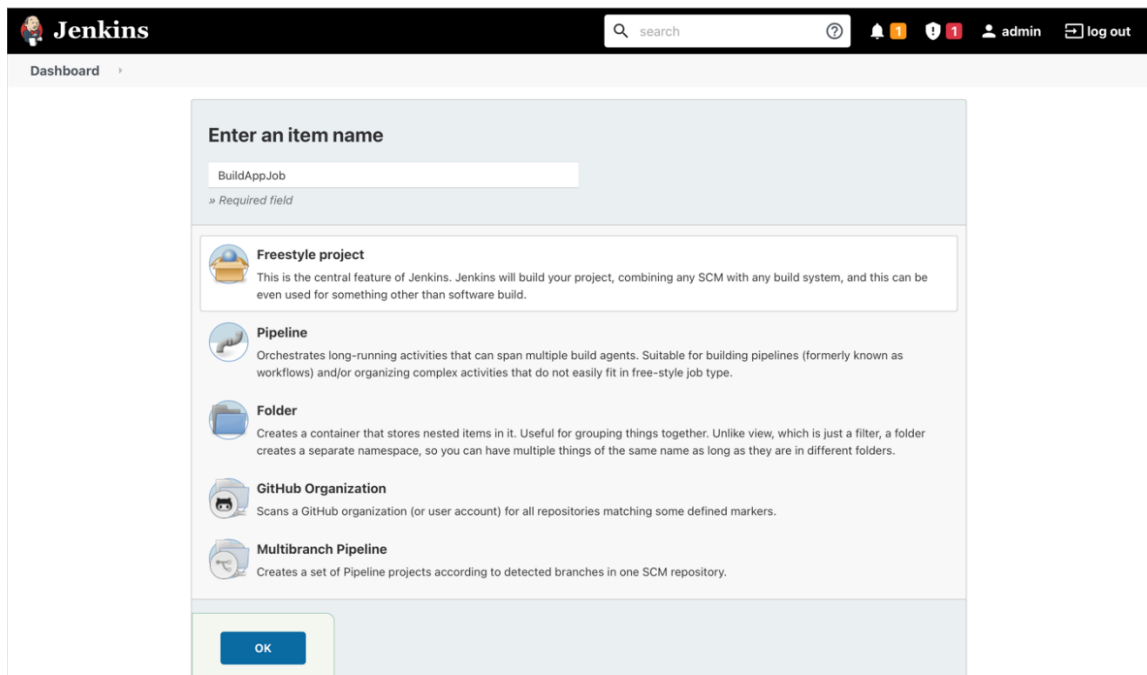
### e) Sử dụng Jenkins để chạy ứng dụng đã dựng

Đơn vị cơ bản của Jenkins là job (hay project), có thể tạo nhiều job với các tác vụ sau:

- Lấy mã nguồn từ repository GitHub
- Build lại ứng dụng bằng script hoặc build tool.
- Đóng gói ứng dụng và chạy nó trên một máy chủ.

#### Bước 1. Tạo job mới

- Chọn *Create a job*
- Trong trường *Enter an item name*, điền **BuildAppJob**
- Chọn thể loại *Freestyle project*
- Chọn *OK*

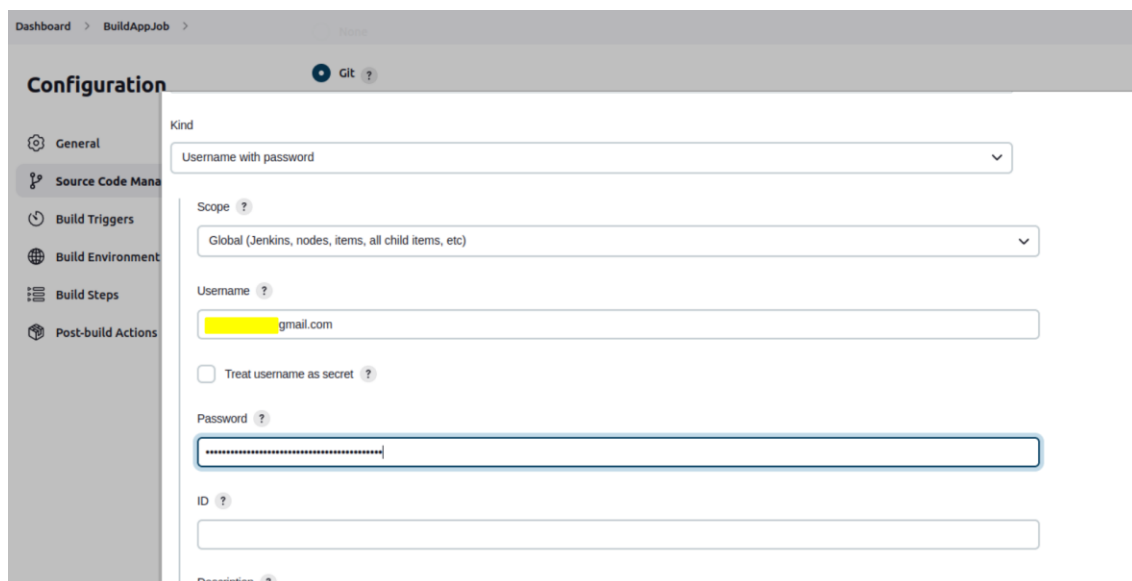


## Bước 2. Cấu hình Jenkins BuildAppJob

Bây giờ ta đang ở cửa sổ cấu hình, nơi ta có thể nhập chi tiết cho job của mình. Các tab trên cùng là phím tắt cho các tùy chọn bên dưới.

- Chọn tab *General*, thêm mô tả cho job. Ví dụ: "**Nhóm X's** first Jenkins job."
- Chọn tab *Source Code Management* và chọn *Git*. Thêm liên kết repository GitHub của ứng dụng. VD: <https://github.com/username/sample-app.git>
- Tại *Credentials*, chọn nút *Add* và chọn *Jenkins*
- Ở cửa sổ *Add Credentials*, điền GitHub username và password (token), chọn *Add*.

*Lưu ý: Token cần được cấu hình gán đủ quyền truy cập để Jenkins sử dụng được.*



- Quay lại dropdown *Credentials* tại tab *General*, chọn cấu hình credential vừa tạo.

**Configuration**

- General
- Source Code Management**
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

**Source Code Management**

☐ None

☒ **Git**

**Repositories**

Repository URL:

Credentials:

- Sau khi ta đã thêm đúng URL và thông tin đăng nhập, Jenkins kiểm tra quyền truy cập vào repository. Nếu có thông báo lỗi (màu đỏ) thì kiểm tra lại bước trên.
- Chọn tab *Build Steps*. Ở dropdown *Add build step*, chọn *Execute shell/* để tạo 1 script thực hiện chạy ứng dụng Sample App.

Dashboard > BuildAppJob >

**Configuration**

- General
- Source Code Management
- Build Triggers
- Build Environment**
- Build Steps
- Post-build Actions

**Build Environment**

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Add timestamps to the Console Output
- ☐ Inspect build log for published Gradle build scans
- ☐ Terminate a build if it's stuck
- ☐ With Ant

**Build Steps**

Filter

- Execute Windows batch command
- Execute shell**
- Invoke Ant
- Invoke Gradle script

- Trong trường *Command*, nhập lệnh ta sử dụng để chạy ứng dụng *sample-app.sh*

Dashboard > BuildAppJob >

**Configuration**

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps**
- Post-build Actions

**Build Steps**

**Execute shell**

Command

See [the list of available environment variables](#)

- Chọn *Save*. Ta quay trở lại Jenkins dashboard với BuildAppJob đã được chọn.

### Bước 3. Dừng Jenkins để build ứng dụng

Lưu ý: trước khi build cần xóa container `samplerunning` đang chạy app `sample-app` trên máy ảo để không bị đùng độ (dùng lệnh `docker stop` và `docker rm`).

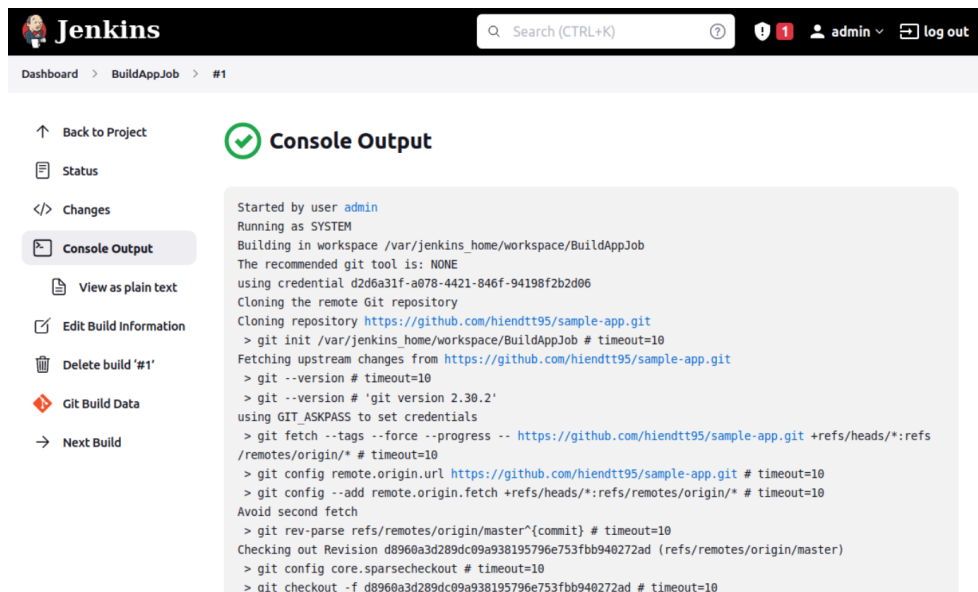
Ở menu bên trái, chọn **Build Now** để bắt đầu job. Jenkins sẽ tải xuống repository Git của chúng ta và thực thi lệnh chạy `bash ./sample-app.sh`.

### Bước 4. Truy cập chi tiết bản dựng

Trong phần *Build History*, chọn 1 bản chạy ứng dụng bất kỳ với ký hiệu `#build number`.

### Bước 5. Xem console output

Chọn *Console Output*.



```

Started by user admin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/BuildAppJob
The recommended git tool is: NONE
using credential d2d6a31f-a078-4421-846f-94198f2b2d06
Cloning the remote Git repository
Cloning repository https://github.com/hientt95/sample-app.git
> git init /var/jenkins_home/workspace/BuildAppJob # timeout=10
Fetching upstream changes from https://github.com/hientt95/sample-app.git
> git --version # timeout=10
> git --version # 'git version 2.30.2'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/hientt95/sample-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/hientt95/sample-app.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision d8960a3d289dc09a938195796e753fbb940272ad (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f d8960a3d289dc09a938195796e753fbb940272ad # timeout=10
  
```

### Báo cáo kết quả trong Console Output?

Thông báo thành công và kết quả lệnh `docker ps -a`. Ở container, ứng dụng vừa dựng chạy ở port 5050 và Jenkins server là 8080.

```

--> 0c8b058e3e6f
Successfully built 0c8b058e3e6f
Successfully tagged sampleapp:latest
0b26a16e1d01747288c68c130b8458529cba6a6cc44916a231c4b28764c532de
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
0b26a16e1d01   sampleapp     "/bin/sh -c 'python ..." Less than a second ago   Up Less than a second
0.0.0.0:5050->5050/tcp, :::5050->5050/tcp   samplerunning
097f6a77887b   jenkins/jenkins:lts "/usr/bin/tini -- /u..." 6 minutes ago   Up 6 minutes
0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 50000/tcp   jenkins_server
Finished: SUCCESS
  
```

**Bước 6.** Mở trình duyệt và truy cập vào <https://localhost:5050> để xác nhận ứng dụng `sampleapp` đã chạy thành công.



### f) Dùng Jenkins để kiểm tra bản build

Trong phần này ta sẽ tạo job thứ 2, kiểm tra ứng dụng hoạt động bình thường.

Lưu ý: Cần đảm bảo đã dừng và xóa docker container samplerunning.

Sinh viên hoàn thành job kiểm tra bản dựng ứng dụng tự động theo 5 bước bên dưới, trình bày step-by-step có minh chứng.

#### Bước 1. Tạo job mới để kiểm tra **sample-app**

Yêu cầu đặt tên: *TestAppJob\_NhomX*

#### Bước 2. Cấu hình Jenkins TestAppJob

- Đặt mô tả: "NhomX's first Jenkins test."
- *Source Code Management* chọn *None*
- Ở *Build Triggers* tab, chọn checkbox *Build after other projects are built* và điền *BuildAppJob* và *Trigger only if build is stable*. Việc này đảm bảo job test này sẽ được kích hoạt *khi và chỉ khi* job build đã thành công.

#### Bước 3. Viết tập lệnh thử nghiệm sẽ chạy sau khi dựng BuildAppJob

- Chọn *Build Steps*, chọn *Add build step* và chọn *Execute shell*.
- Yêu cầu viết bash shell:
  - o Truy cập đường dẫn ứng dụng.
  - o Nhận kết quả trả về.
  - o Kiểm tra trong đó có tồn tại thông tin nhận biết thành công hay không. Nếu thành công thì exit code là 0 và ngược lại là 1.
- Chọn *Save*

#### Bước 4. Yêu cầu Jenkins chạy lại job BuildAppJob

#### Bước 5. Kiểm tra cả 2 job hoàn thành hay không

Nếu thành công, trong Dashboard ta sẽ thấy cột Last Success ở cả BuildAppJob và TestAppJob\_NhomX.

### g) Tạo Pipeline trong Jenkins

Mặc dù có thể chạy hai job của mình bằng cách chỉ cần nhấp vào nút **Build Now** cho BuildAppJob, nhưng các dự án phát triển phần mềm thường phức tạp hơn nhiều. Những dự án này có thể được hưởng lợi rất nhiều từ việc tự động hóa các bản build để tích hợp liên tục các thay đổi đoạn mã và liên tục tạo các bản dựng phát triển đã sẵn sàng để triển khai. Đây là bản chất của CI/CD. Một pipeline có thể được tự động hóa điều này.

Sinh viên hoàn thành pipeline của 2 ứng dụng Buid và Test theo gợi ý bên dưới, trình bày step-by-step có minh chứng.

**Bước 1.** Tạo job Pipeline

- Chọn *New Item*
- Ở trường *Enter an item name*, điền ***SamplePipeline\_NhomX***
- Chọn kiểu job Pipeline
- Chọn *OK*

**Bước 2.** Cấu hình job SamplePipeline\_NhomX

- Ở tab *Pipeline*, trong phần *Script* điền đoạn mã sau:

```
node {
  stage('Preparation') {
    catchError(buildResult: 'SUCCESS') {
      sh 'docker stop samplerunning'
      sh 'docker rm samplerunning'
    }
  }
  stage('Build') {
    build 'BuildAppJob'
  }
  stage('Results') {
    build 'TestAppJob_NhomX'
  }
}
```

**Sinh viên giải thích đoạn mã trên?**

- Chọn *Save* và ta sẽ trở về Jenkins dashboard của SamplePipeline

**Bước 3.** Chạy SamplePipeline.

Chọn *Buid Now* để chạy job. Nếu code lỗi thì vào *Stage View* để xem log.

**Bước 4.** Kiểm tra SamplePipeline output

Ở mục *Build History*, chọn bản build với ký hiệu *#build number*, tiếp tục chọn *Console Output*.

---

***Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này***

## YÊU CẦU CHUNG

- Sinh viên tìm hiểu và thực hiện bài tập theo yêu cầu, hướng dẫn.
- Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

### Báo cáo:

- File **.PDF**. Tập trung vào nội dung thực hiện, không mô tả lý thuyết.
- Đặt tên theo định dạng: **[Mã lớp]-AssignmentX\_NhomY**. (trong đó X là Thứ tự Bài tập, Y là số thứ tự nhóm trong danh sách nhóm đồ án).

Ví dụ: **[NT521.012.ATCL]-Assignment3\_Nhom03**.

- Nếu báo cáo có nhiều file, nén tất cả file vào file **.ZIP** với cùng tên file báo cáo.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại [courses.uit.edu.vn](https://courses.uit.edu.vn).

### Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

*Bài sao chép, trễ, ... sẽ được xử lý tùy mức độ vi phạm.*

**HẾT**