

BÁO CÁO THỰC HÀNH

LAB 04: Tìm hiểu về Docker

Họ và tên: Lê Hoàng Khang

Mã số sinh viên: 21083791

1 Phần 1: Các lệnh cơ bản thao tác với Docker

1.1 docker --version

```
C:\Users\iamho>docker version
Client:
Version:           28.0.1
API version:       1.48
Go version:        go1.23.6
Git commit:        068a01e
Built:             Wed Feb 26 10:41:52 2025
OS/Arch:           windows/amd64
Context:           desktop-linux

Server: Docker Desktop 4.39.0 (184744)
Engine:
Version:           28.0.1
API version:       1.48 (minimum version 1.24)
Go version:        go1.23.6
Git commit:        bbd0a17
Built:             Wed Feb 26 10:41:16 2025
OS/Arch:           linux/amd64
Experimental:      false
containerd:
Version:           1.7.25
GitCommit:         bcc810d6b9066471b0b6fa75f557a15a1cbf31bb
runc:
Version:           1.2.4
GitCommit:         v1.2.4-0-g6c52b3f
docker-init:
Version:           0.19.0
GitCommit:         de40ad0
```

Mô tả : Kiểm tra phiên bản của Docker hiện đang được cài đặt trên hệ thống.

Kết quả : Hiển thị phiên bản Docker

1.2 docker run hello-world

```
C:\Users\iamho>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b010412b6ea13de1efb19
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Mô tả : Chạy một container từ image hello-world. Đây là một image đơn giản dùng để kiểm tra xem Docker có hoạt động đúng không.

Quá trình :

Nếu image hello-world chưa tồn tại cục bộ, Docker sẽ tải nó từ Docker Hub.

Sau đó, container sẽ chạy và in ra thông báo "Hello from Docker!".

1.3 docker pull nginx

```
C:\Users\iamho>docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
6e909acdb790: Already exists
5eaa34f5b9c2: Pull complete
417c4bccf534: Pull complete
e7e0ca015e55: Pull complete
373fe654e984: Pull complete
97f5c0f51d43: Pull complete
c22eb46e871a: Pull complete
Digest: sha256:57a563126c0fd426346b02e5aa231ae9e5fd66f2248b36553207a0eca1403fde
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

Mô tả : Tải xuống (pull) image nginx từ Docker Hub mà không chạy nó.

Kết quả : Image nginx sẽ được lưu trữ cục bộ trên máy tính của bạn.

1.4 docker images

```
C:\Users\iamho>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
edu-forge-assessment-api	latest	48b53a804791	41 hours ago	415MB
postgres	17	76e3e031d245	2 weeks ago	438MB
nginx	latest	53a18edff809	5 weeks ago	192MB
hello-world	latest	74cc54e27dc4	7 weeks ago	10.1kB

Mô tả : Liệt kê tất cả các Docker images hiện có trên hệ thống.

Kết quả : Hiển thị danh sách các images với thông tin như REPOSITORY, TAG, IMAGE ID, CREATED, và SIZE.

1.5 docker run -d nginx

```
C:\Users\iamho>docker run -d nginx
ebec5202fc89d83a16fb1a415f16450c1008d61e479f0833248b5e48c0af26f4
```

Mô tả: Chạy một container từ image nginx ở chế độ nền (detached mode).

Tham số -d : Container sẽ chạy ở chế độ nền mà không hiển thị log ra terminal.

Kết quả: Một container mới sẽ được khởi tạo và chạy ở nền.

1.6 docker ps

```
C:\Users\iamho>docker run -d nginx
ebec5202fc89d83a16fb1a415f16450c1008d61e479f0833248b5e48c0af26f4
```

Mô tả: Liệt kê các container đang chạy hiện tại.

Kết quả: Hiển thị danh sách các container đang hoạt động với thông tin như CONTAINER ID, IMAGE, STATUS, PORTS, và NAMES.

1.7 docker ps -a

```
C:\Users\iamho>docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ebec5202fc89	nginx	"/docker-entrypoint..."	3 minutes ago	Up 3 minutes	80/tcp	busy_ptolemy
67943558c9b8	nginx	"/docker-entrypoint..."	4 minutes ago	Exited (127) 4 minutes ago		exciting_kirch
86ca78d413de	hello-world	"/hello"	12 minutes ago	Exited (0) 12 minutes ago		thirsty_banach
5ec9af2ebbd0	postgres:17	"docker-entrypoint.s..."	6 hours ago	Exited (0) 6 hours ago		edu-forge-course-db

Mô tả: Liệt kê tất cả các container (bao gồm cả đang chạy và đã dừng).

Kết quả: Hiển thị danh sách đầy đủ các container trên hệ thống.

1.8 docker logs <container_id>

```
C:\Users\iamho>docker logs 5ec9af2ebbd0
PostgreSQL Database directory appears to contain a database; Skipping initialization

2025-03-18 03:14:12.655 UTC [1] LOG: starting PostgreSQL 17.4 (Debian 17.4-1.pgdg120+2) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
2025-03-18 03:14:12.655 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
2025-03-18 03:14:12.655 UTC [1] LOG: listening on IPv6 address "::", port 5432
2025-03-18 03:14:12.660 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2025-03-18 03:14:12.706 UTC [29] LOG: database system was shut down at 2025-03-18 03:10:05 UTC
2025-03-18 03:14:12.755 UTC [1] LOG: database system is ready to accept connections
2025-03-18 03:19:12.743 UTC [27] LOG: checkpoint starting: time
2025-03-18 03:19:12.906 UTC [27] LOG: checkpoint complete: wrote 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.024 s, sync=0.007 s, total=0.163 s; sync files=2, longest=0.004 s, average=0.004 s; distance=0 kB, estimate=0 kB; lsn=0/1A07A30, redo lsn=0/1A079D8
2025-03-18 03:19:16.192 UTC [1] LOG: received fast shutdown request
2025-03-18 03:19:16.196 UTC [1] LOG: aborting any active transactions
2025-03-18 03:19:16.197 UTC [1] LOG: background worker "logical replication launcher" (PID 32) exited with exit code 1
2025-03-18 03:19:16.198 UTC [27] LOG: shutting down
2025-03-18 03:19:16.202 UTC [27] LOG: checkpoint starting: shutdown immediate
2025-03-18 03:19:16.248 UTC [27] LOG: checkpoint complete: wrote 0 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.009 s, sync=0.001 s, total=0.050 s; sync files=0, longest=0.000 s, average=0.000 s; distance=0 kB, estimate=0 kB; lsn=0/1A07AA8, redo lsn=0/1A07AA8
2025-03-18 03:19:16.265 UTC [1] LOG: database system is shut down
```

Mô tả: Xem log đầu ra của một container cụ thể (được xác định bằng <container_id> hoặc tên).

Kết quả: Hiển thị các thông điệp log mà container đã ghi lại trong quá trình hoạt động.

1.9 docker exec -it <container_id> /bin/sh

```
C:\Users\iamho>docker exec -it ebec5202fc89 /bin/sh
# cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
# ls
bin  dev  docker-entrypoint.sh  home  lib64  mnt  proc  run  srv  tmp  var
boot  docker-entrypoint.d  etc  lib  media  opt  root /sbin  sys  usr
```

Mô tả: Truy cập vào shell của một container đang chạy để thực hiện các lệnh tương tác.

Tham số:

-it: Kết hợp giữa chế độ tương tác (-i) và thiết bị đầu cuối (-t).

/bin/sh: Chỉ định shell sẽ sử dụng bên trong container.

Kết quả: Bạn có thể thực thi các lệnh trực tiếp trong container

1.10 docker stop <container_id>

```
C:\Users\iamho>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
ebec5202fc89   nginx    "/docker-entrypoint..." 9 minutes ago  Up 9 minutes  80/tcp       busy_ptolemy

C:\Users\iamho>docker stop ebec5202fc89
ebec5202fc89

C:\Users\iamho>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
```

Mô tả: Dừng một container đang chạy.

Kết quả: Container sẽ dừng hoạt động nhưng vẫn tồn tại trên hệ thống.

1.11 docker restart <container_id>

```
C:\Users\iamho>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
be4dd9946546   nginx    "/docker-entrypoint..." 5 seconds ago  Up 4 seconds  80/tcp       admiring_zhukovsky

C:\Users\iamho>docker restart be4dd9946546
be4dd9946546

C:\Users\iamho>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
be4dd9946546   nginx    "/docker-entrypoint..." 20 seconds ago  Up 3 seconds  80/tcp       admiring_zhukovsky
```

Mô tả: Khởi động lại một container.

Kết quả: Container sẽ được dừng và khởi động lại ngay lập tức.

1.12 docker rm <container_id>

```
C:\Users\iamho>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
be4dd9946546   nginx    "/docker-entrypoint..." 20 seconds ago  Up 3 seconds  80/tcp       admiring_zhukovsky

C:\Users\iamho>docker rm be4dd9946546
Error response from daemon: cannot remove container "/admiring_zhukovsky": container is running: stop the container before removing or force remove

C:\Users\iamho>docker stop be4dd9946546
be4dd9946546

C:\Users\iamho>docker rm be4dd9946546
be4dd9946546

C:\Users\iamho>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
```

Mô tả: Xóa một container đã dừng.

Kết quả: Container bị xóa hoàn toàn khỏi hệ thống.

1.13 docker container prune

```
C:\Users\iamho>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
ebec5202fc89   nginx    "/docker-entrypoint...." 14 minutes ago Exited (0) 4 minutes ago
67943558c9b8   nginx    "/docker-entrypoint...." 15 minutes ago Exited (127) 15 minutes ago
86ca78d413de   hello-world "/hello"                24 minutes ago Exited (0) 24 minutes ago
5ec9af2ebbd0   postgres:17 "docker-entrypoint.s..." 6 hours ago    Exited (0) 6 hours ago
edu-forge-course-db

C:\Users\iamho>docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
ebec5202fc89d83a16fb1a415f16450c1008d61e479f0833248b5e48c0af26f4
67943558c9b8025ba87648ac5bfcfc46b37d75d4ccea9cb6356203494f14ce95
86ca78d413de550818b6c70b404fd214dc0cde605b29db783259ba1be7799b1e
5ec9af2ebbd03c61f32091184d57d8f4dae06e6489343f8452d027e7e0b2c0eb

Total reclaimed space: 1.093kB

C:\Users\iamho>docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
```

Mô tả: Xóa tất cả các container đã dừng.

Kết quả: Giải phóng tài nguyên bằng cách xóa các container không còn cần thiết.

1.14 docker rmi <image_id>

```
C:\Users\iamho>docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
edu-forge-assessment-api latest      48b53a804791 41 hours ago  415MB
postgres            17         76e3e031d245 2 weeks ago   438MB
nginx               latest     53a18edff809 5 weeks ago   192MB
hello-world         latest     74cc54e27dc4 7 weeks ago   10.1kB

C:\Users\iamho>docker rmi hello-world
Untagged: hello-world:latest
Untagged: hello-world@sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b010412b6ea13de1efb19
Deleted: sha256:74cc54e27dc41bb10dc4b2226072d469509f2f22f1a3ce74f4a59661a1d44602
Deleted: sha256:63a41026379f4391a306242eb0b9f26dc3550d863b7fd8b97d899f6eb89efe72

C:\Users\iamho>docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
edu-forge-assessment-api latest      48b53a804791 41 hours ago  415MB
postgres            17         76e3e031d245 2 weeks ago   438MB
nginx               latest     53a18edff809 5 weeks ago   192MB
```

1.15 docker image prune -a

```
C:\Users\iamho>docker images
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
edu-forge-assessment-api  latest         48b53a804791    41 hours ago    415MB
postgres            17             76e3e031d245    2 weeks ago     438MB
nginx               latest         53a18edff809    5 weeks ago     192MB

C:\Users\iamho>docker image prune -a
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: postgres:17
deleted: sha256:58bb12171087c66904a8e9ef17ddfd8e36580ffa3926cb15b4e212abf3f7ef30
deleted: sha256:76e3e031d245f3b1018ac3d56a65130a0b7b6c2a55c16b856445a9fb04ddefaf
deleted: sha256:34dba13653a154d83cdf340ed91223f56e172ebfa1e049c7bbe139fbde3314ac
deleted: sha256:5322758e58121b840581203869db9f41541b596864b18a775176d3079765ff3e
deleted: sha256:72eb08b8dafd38c9ebff32ba992900793b63ae1bb8e403c3952b970a25b3a74f
deleted: sha256:bb1249932f5406c6477dff791e1461a1475670bd8fac41623fa59b6c4df20c68
deleted: sha256:ab1a811cfa30750297aff9cd6efac0b875d75f5cbf4c2d4f7fe457762d2e01d9
deleted: sha256:e76f6f3486210f50aff097c536d23458943a44b3222e10fff4833e768cfb3188b
deleted: sha256:58fe591ba71e23c5d369016b9babb2371dc6906565dfb28b1a3ba46a70efa9fc
deleted: sha256:5a70e2eed643e5a976e6e955990656921aa20d147eff2d013eef6bb6e84f348c
deleted: sha256:ea5b9d404e6192310cabbac664c01ef6d421a08128dec2186785f47ec9ebf801
deleted: sha256:3c3f85bd525f63870a32ee31b179bfa891844b969bfe95340b15c932e94fe6de
deleted: sha256:6cabe81fae96d69fe24f67aee0479d832c0ae68d9639e5753a6f2846b2f99d5a
deleted: sha256:66dda6fe1fe2795fac67649338a0278408c89530247ae1c3a76d3e5af02ca347
deleted: sha256:942e092fb5f3fec51cdb3772bb8f74b79c8a4f202f94135838151c13d0129926
untagged: nginx:latest
deleted: sha256:57a563126c0fd426346b02e5aa231ae9e5fd66f2248b36553207a0eca1403fde
deleted: sha256:53a18edff8091d5faf1e42b4d885bc5f0f897873b0b8f0ace236cd5930819b0
deleted: sha256:9624c14fde1debd1256228b54278fec5e576a42dcbf73f420762a91f4a06c87
deleted: sha256:75cef3a8c4e762e0d3d0c01f5e5cf9407478057005f945fa78edef29a2bc6e33
deleted: sha256:bff22610f6a6c90cb4a456617b926c87cb1c50efd3f90b1d96d9c88e5f4b75a6e
deleted: sha256:8e41d2be566aeafda18718a8a4b8c515c50b06f82cd7a92420ae91010773e15c
deleted: sha256:da2d6794d8696a98178b6882353953c9f410dcffff428cfa3caa5759036d24bd
deleted: sha256:e9228041e2928859e124edaf5a456926097605092e1855d51aa9e43f984f770e
deleted: sha256:1287fbecdfc6e6ee8cf2436e5b9e9d86a4648db2d91080377d499737f1b307f3
untagged: edu-forge-assessment-api:latest
deleted: sha256:48b53a804791d6e7d1bb36fa8909ec002d0bd01ec85fd2470252ee592a60f9a6

Total reclaimed space: 555.6MB

C:\Users\iamho>docker images
REPOSITORY          TAG             IMAGE ID         CREATED          SIZE
```

Mô tả: Xóa một image cụ thể.

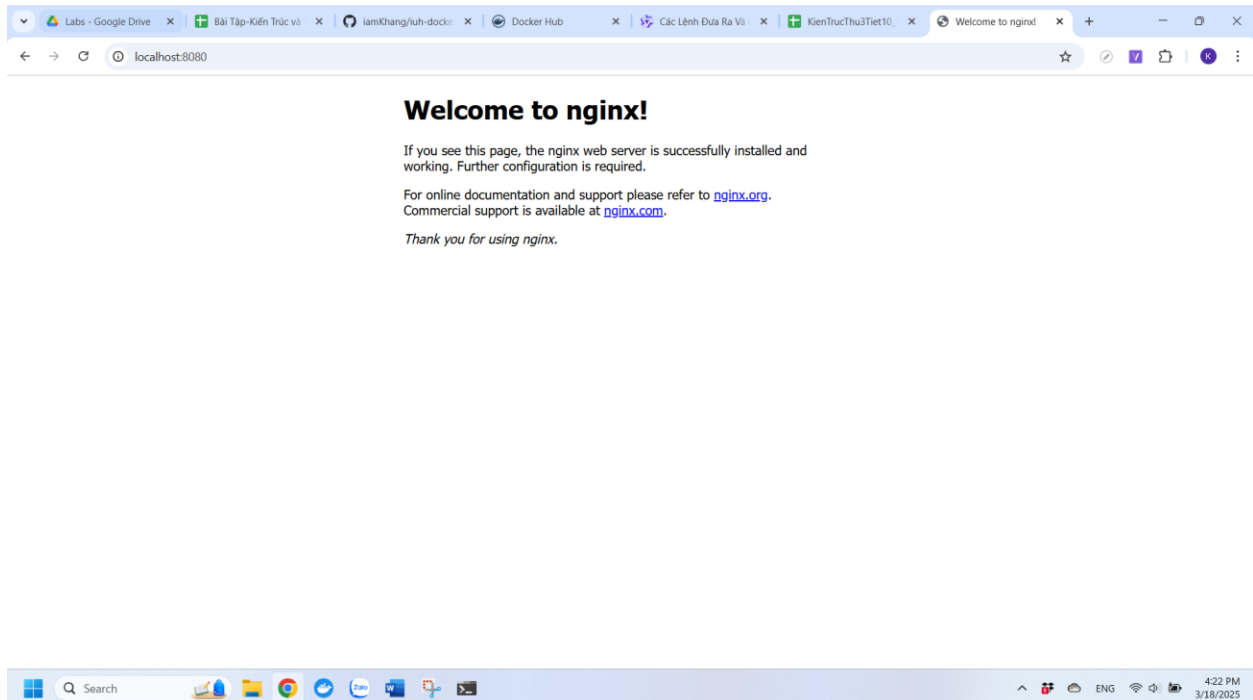
Lưu ý: Image chỉ có thể bị xóa nếu không có container nào đang sử dụng nó.

1.16 docker run -d -p 8080:80 nginx

```
C:\Users\iamho>docker run -d -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
6e909acdb790: Pull complete
5eaa34f5b9c2: Pull complete
417c4bccf534: Pull complete
e7e0ca015e55: Pull complete
373fe654e984: Pull complete
97f5c0f51d43: Pull complete
c22eb46e871a: Pull complete
Digest: sha256:57a563126c0fd426346b02e5aa231ae9e5fd66f2248b36553207a0eca1403fde
Status: Downloaded newer image for nginx:latest
a78ba98876b4b00c68cb9c9db31ec357f2e3dc11287fa0090a3c9f2705555909
```

Mô tả: Chạy một container từ image nginx ở chế độ nền và ánh xạ cổng 8080 trên host tới cổng 80 của container.

Kết quả: Bạn có thể truy cập ứng dụng Nginx bằng cách mở trình duyệt và nhập địa chỉ <http://localhost:8080>.



1.17 docker inspect <container_id>

```
C:\Users\iamho>docker inspect a78ba98876b4
[
  {
    "Id": "a78ba98876b4b00c68cb9c9db31ec357f2e3dc11287fa0090a3c9f2705555909",
    "Created": "2025-03-18T09:21:18.577432845Z",
    "Path": "/docker-entrypoint.sh",
    "Args": [
      "nginx",
      "-g",
      "daemon off;"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 1174,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2025-03-18T09:21:18.962876317Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:53a18edff8091d5faaff1e42b4d885bc5f0f097873b0b8f8ace236cd5930819b0",
    "ResolvConfPath": "/var/lib/docker/containers/a78ba98876b4b00c68cb9c9db31ec357f2e3dc11287fa0090a3c9f2705555909/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/a78ba98876b4b00c68cb9c9db31ec357f2e3dc11287fa0090a3c9f2705555909/hostname",
    "HostsPath": "/var/lib/docker/containers/a78ba98876b4b00c68cb9c9db31ec357f2e3dc11287fa0090a3c9f2705555909/hosts",
    "LogPath": "/var/lib/docker/containers/a78ba98876b4b00c68cb9c9db31ec357f2e3dc11287fa0090a3c9f2705555909/a78ba98876b4b00c68cb9c9db31ec357f2e3dc11287fa0090a3c9f270555909-json.log",
    "Name": "/wizardly_jennings",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,
      "ContainerIDFile": "",

```

Mô tả: Xem thông tin chi tiết về một container (hoặc image).

Kết quả: Hiển thị thông tin cấu hình, mạng, và trạng thái của container dưới dạng JSON.

1.18 docker run -d -v mydata:/data nginx

```
C:\Users\iamho>docker run -d -v mydata:/data nginx
081358cae8fe79c69ea64dfa4de0da5f1f418ed531d40be6aa3599a782c1f563
```

Mô tả: Chạy một container từ image nginx và gắn một volume có tên mydata vào đường dẫn /data trong container.

Kết quả: Dữ liệu trong thư mục /data của container sẽ được lưu trữ bền vững trong volume mydata.

1.19 docker volume ls

```
C:\Users\iamho>docker volume ls
DRIVER      VOLUME NAME
local       edu-forge-assessment-postgres_data
local       mydata
```

Mô tả: Liệt kê tất cả các volumes hiện có trên hệ thống.

Kết quả: Hiển thị danh sách các volumes cùng với thông tin liên quan.

1.20 docker volume prune

```
C:\Users\iamho>docker volume prune
WARNING! This will remove anonymous local volumes not used by at least one container.
Are you sure you want to continue? [y/N] y
Total reclaimed space: 0B
```

Mô tả: Xóa tất cả các volumes không được sử dụng bởi bất kỳ container nào.

Kết quả: Giải phóng không gian đĩa bằng cách loại bỏ các volumes dư thừa.

1.21 docker run -d --name my_nginx nginx

```
C:\Users\iamho>docker run -d --name my_nginx nginx
438d02edc374ae895e13dda97660b5d7ee52fb243b955b5cd95792a92181e5c8
```

Mô tả: Chạy một container từ image nginx ở chế độ nền và đặt tên cho container là my_nginx.

Kết quả: Container có tên dễ nhớ để quản lý.

1.22 docker stats

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
438d02edc374	my_nginx	0.00%	14.07MiB / 7.598GiB	0.18%	872B / 126B	0B / 0B	17
081358cae8fe	suspicious_dubinsky	0.00%	15.48MiB / 7.598GiB	0.20%	998B / 126B	0B / 0B	17
a78ba98876b4	wizardly_jennings	0.00%	12.85MiB / 7.598GiB	0.17%	4.01kB / 2.95kB	0B / 0B	17

Mô tả: Hiển thị thông tin thời gian thực về việc sử dụng tài nguyên (CPU, memory, network, I/O) của các container đang chạy.

Kết quả: Bảng thống kê tài nguyên theo thời gian thực.

1.23 docker network ls

```
C:\Users\iamho>docker network ls
NETWORK ID          NAME                                DRIVER              SCOPE
0777ded5b61e        bridge                            bridge              local
8242f7c5a7c1        edu-forge-assessment_edu-forge-network  bridge              local
92bf687269be        hoangkhang-net                  bridge              local
271ea6b66cf5        host                             host                local
dcda001b81a9        none                             null                local
```

Mô tả: Liệt kê tất cả các mạng Docker hiện có.

Kết quả: Hiển thị danh sách các mạng cùng với thông tin như NETWORK ID, NAME, và DRIVER.

1.24 docker network create my_network

```
C:\Users\iamho>docker network create my_network
7e7e1e0c102dfec3e6526dd814d31d188a34f727d75760028509263639226460
```

Mô tả: Tạo một mạng Docker mới có tên my_network.

Kết quả: Mạng mới được tạo để kết nối các container.

1.25 docker run -d --network my_network --name my_container nginx

```
C:\Users\iamho>docker run -d --network my_network --name my_container nginx
9d5e9c8423e81ab63e07e6390b9f385d2e2cd3881eed117c374e0a611743ced8
```

Mô tả: Chạy một container từ image nginx và gắn nó vào mạng my_network với tên my_container.

Kết quả: Container có thể giao tiếp với các container khác trong cùng mạng.

1.26 docker network connect my_network my_nginx

```
C:\Users\iamho>docker network connect my_network my_nginx
```

```
C:\Users\iamho>docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
0777ded5b61e	bridge	bridge	local
8242f7c5a7c1	edu-forge-assessment_edu-forge-network	bridge	local
92bf687269be	hoangkhang-net	bridge	local
271ea6b66cf5	host	host	local
7e7e1e0c102d	my_network	bridge	local
dcda001b81a9	none	null	local

Mô tả: Kết nối một container hiện có (my_nginx) vào mạng my_network.

Kết quả: Container có thể giao tiếp với các thành viên khác trong mạng.

1.27 docker run -d -e MY_ENV=hello_world nginx

```
C:\Users\iamho>docker run -d -e MY_ENV=hello_world nginx
981392fcb998d7a825a9f39d90d879e7c0155c939ed88fbce743c530bc58a8be
```

Mô tả: Chạy một container từ image nginx và thiết lập biến môi trường MY_ENV với giá trị hello_world.

Kết quả: Biến môi trường có thể được sử dụng bên trong container.

1.28 docker logs -f my_nginx

```
C:\Users\iamho>docker logs -f my_nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/03/18 09:25:40 [notice] 1#1: using the "epoll" event method
2025/03/18 09:25:40 [notice] 1#1: nginx/1.27.4
2025/03/18 09:25:40 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/03/18 09:25:40 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/03/18 09:25:40 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/03/18 09:25:40 [notice] 1#1: start worker processes
2025/03/18 09:25:40 [notice] 1#1: start worker process 28
2025/03/18 09:25:40 [notice] 1#1: start worker process 29
2025/03/18 09:25:40 [notice] 1#1: start worker process 30
2025/03/18 09:25:40 [notice] 1#1: start worker process 31
2025/03/18 09:25:40 [notice] 1#1: start worker process 32
2025/03/18 09:25:40 [notice] 1#1: start worker process 33
2025/03/18 09:25:40 [notice] 1#1: start worker process 34
2025/03/18 09:25:40 [notice] 1#1: start worker process 35
2025/03/18 09:25:40 [notice] 1#1: start worker process 36
2025/03/18 09:25:40 [notice] 1#1: start worker process 37
2025/03/18 09:25:40 [notice] 1#1: start worker process 38
2025/03/18 09:25:40 [notice] 1#1: start worker process 39
2025/03/18 09:25:40 [notice] 1#1: start worker process 40
2025/03/18 09:25:40 [notice] 1#1: start worker process 41
2025/03/18 09:25:40 [notice] 1#1: start worker process 42
2025/03/18 09:25:40 [notice] 1#1: start worker process 43
```

Mô tả: Xem log của container my_nginx theo thời gian thực.

Tham số -f : Theo dõi log liên tục (giống như lệnh tail -f).

1.29 Tạo file với nội dung:

FROM nginx

COPY index.html /usr/share/nginx/html/index.html

```
D:\my_nginx_project>copy con Dockerfile
FROM nginx
COPY index.html /usr/share/nginx/html/index.html
^Z

1 file(s) copied.
```

```
D:\my_nginx_project>echo "<h1>Hello from my custom Nginx image!</h1>" > index.html
```

1.30 docker build -t my_nginx_image .

```
D:\>docker build -t my_nginx_image .
[+] Building 0.1s (1/1) FINISHED
=> [internal] load build definition from Dockerfile
=> == transferring dockerfile: 2B
ERROR: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory
View build details: docker-desktop:///dashboard/build/desktop-linux/desktop-linux/scmytnwli8qocpt9lto6ip8j
```

Build docker image với nội dung vừa tạo

1.31 docker run -d -p 8080:80 my_nginx_image

```
D:\my_nginx_project>docker build -t my_nginx_image .
[+] Building 0.2s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 108B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 84B
=> [1/2] FROM docker.io/library/nginx:latest
=> [2/2] COPY index.html /usr/share/nginx/html/index.html
=> exporting to image
=> => exporting layers
=> => writing image sha256:d7c0f89c2ae92fb2ddc978cd7fd792b6ed67185fa7b240d68cfd734c54fc8536
=> => naming to docker.io/library/my_nginx_image
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/umsekyutkqm54xsqrbv4ro27
```

Chạy docker image vừa tạo ở bước trên

Bài 1: Tạo Dockerfile chạy một ứng dụng Node.js đơn giản

Các bước thực hiện:

Bước 1: Tạo file app.js và Dockerfile

Bước 2: Chạy lệnh docker build -t nodejs-docker-app .

```
E:\iuh-docker\nodejs_docker_app>docker build -t nodejs-docker-app .
[+] Building 2.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 413B
=> [internal] load metadata for docker.io/library/node:18
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:18@sha256:15f001804eb79fdc82f9704b44e0d3f14329b50df8c36da011523e7480564d8d
=> [internal] load build context
=> => transferring context: 312B
=> CACHED [2/5] WORKDIR /app
=> [3/5] COPY package*.json ./
=> [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:447cfe4586b0cfb4e4a7a370ead67302866832c896f11c88a521a7b7073c91b9
=> => naming to docker.io/library/nodejs-docker-app
```

Bước 3: Chạy lệnh docker run -d -p 3000:3000 --name nodejs_container nodejs-docker-app

```
E:\iuh-docker\nodejs_docker_app>docker run -d -p 3000:3000 --name nodejs_container nodejs-docker-app
aed3981646790e1da7ae11b5b81233df86e6af8c9e05512a2fa1bbac378d41d2
```

Bước 4: Truy cập <http://localhost:3000> để xem kết quả

Bài 2: Tạo Dockerfile chạy một ứng dụng Python Flask

Yêu cầu:

Viết Dockerfile để chạy một ứng dụng Flask hiển thị "Hello, Docker Flask!" trên cổng 5000.

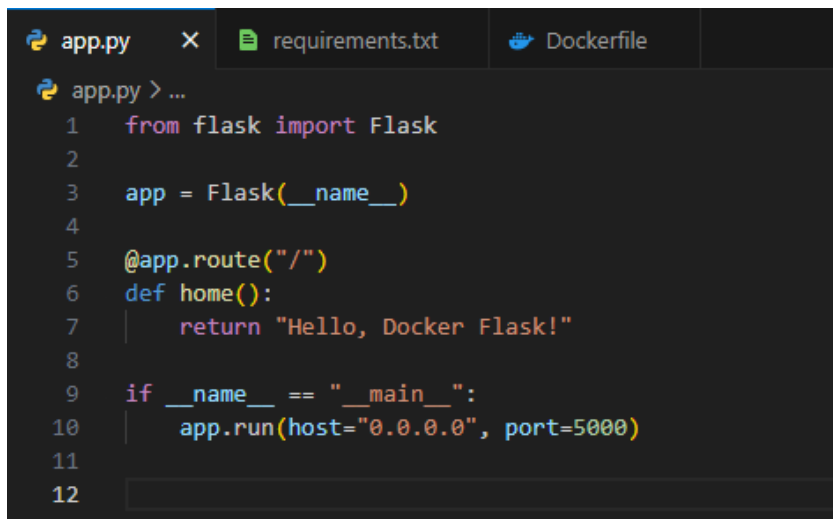
Sử dụng python:3.9 làm base image.

Bài 2: Tạo Dockerfile chạy một ứng dụng Python Flask

Bước 1: Tạo các file cần thiết

```
E:\iuh-docker>mkdir flask-docker  
  
E:\iuh-docker>cd flask-docker  
  
E:\iuh-docker\flask-docker>echo. > app.py  
  
E:\iuh-docker\flask-docker>echo. > requirements.txt  
  
E:\iuh-docker\flask-docker>echo. > Dockerfile
```

Bước 2: Viết ứng dụng và cấu hình docker file



The screenshot shows a code editor with three tabs: app.py, requirements.txt, and Dockerfile. The app.py tab is active, displaying the following Python code:

```
1 from flask import Flask  
2  
3 app = Flask(__name__)  
4  
5 @app.route("/")  
6 def home():  
7     return "Hello, Docker Flask!"  
8  
9 if __name__ == "__main__":  
10     app.run(host="0.0.0.0", port=5000)  
11  
12
```

```
app.py requirements.txt Dockerfile X
Dockerfile > ...
1  # Sử dụng Python 3.9 làm base image
2  FROM python:3.9
3
4  # Đặt thư mục làm việc trong container
5  WORKDIR /app
6
7  # Sao chép file vào container
8  COPY . .
9
10 # Cài đặt thư viện cần thiết
11 RUN pip install -r requirements.txt
12
13 # Mở cổng 5000
14 EXPOSE 5000
15
16 # Chạy ứng dụng Flask
17 CMD ["python", "app.py"]
18
```

Bước 3: Chạy lệnh build ứng dụng và start container

`docker build -t flask-app .`

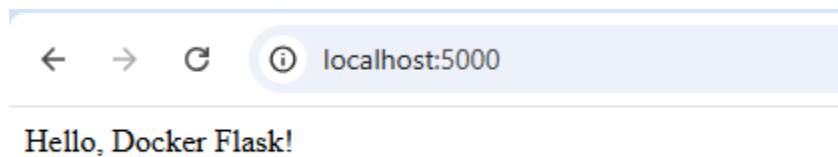
```
PS E:\iuh-docker\flask-docker> docker build -t flask-app .
[+] Building 28.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 382B
=> [internal] load metadata for docker.io/library/python:3.9
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/python:3.9@sha256:bc2e05bca883473050fc3b7c134c28ab822be73126ba1ce29517d9e8b7f3703b
=> => resolve docker.io/library/python:3.9@sha256:bc2e05bca883473050fc3b7c134c28ab822be73126ba1ce29517d9e8b7f3703b
=> => sha256:3db46eeb095dbddf83794896233f9ed627746b394587600bf4e8e4c68303a915 2.32kB / 2.32kB
=> => sha256:f6d72b00ae7cbea513baa839d4f1bcebb51c434df9602410dfd34bc71e233c8e 6.16MB / 6.16MB
=> => sha256:6e02a90e58aec58d3d8f4549cb6a82a2ccc1db075b8a26a48fe1d0f065b52d86 19.85MB / 19.85MB
=> => sha256:f299e06712452fd49405fe52fb66dc0bbdd14cc8d8342baa8b2741df89dd465d 250B / 250B
=> => sha256:bc2e05bca883473050fc3b7c134c28ab822be73126ba1ce29517d9e8b7f3703b 10.35kB / 10.35kB
=> => sha256:859d4a0f1fd8b03e685b4f3b6f0abfeeb84cca3047f9b87d0a8c8f3c90764365 6.17kB / 6.17kB
=> => extracting sha256:f6d72b00ae7cbea513baa839d4f1bcebb51c434df9602410dfd34bc71e233c8e
=> => extracting sha256:6e02a90e58aec58d3d8f4549cb6a82a2ccc1db075b8a26a48fe1d0f065b52d86
=> => extracting sha256:f299e06712452fd49405fe52fb66dc0bbdd14cc8d8342baa8b2741df89dd465d
=> [internal] load build context
=> => transferring context: 662B
=> [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN pip install -r requirements.txt
=> => exporting to image
=> => exporting layers
=> => writing image sha256:2ca2dc43f9c254b8471a560c00327735b04bd8b9c29c2e99c8f0bfd58ddf5f73
=> => naming to docker.io/library/flask-app
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/xxmu0w1xwe06djaoxf7cm2s2l
```

`docker run -p 5000:5000 flask-app`

```
PS E:\iuh-docker\flask-docker> docker run -p 5001:5001 flask-app
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

(Cảnh báo để cho chúng ta biết việc sử dụng lệnh này không nên dùng trong môi trường Product)

Kết quả



Bài 3: Chạy 1 ứng dụng React đơn giản bằng docker file


```
● PS E:\iuh-docker> npx create-react-app react-docker-app
Need to install the following packages:
create-react-app@5.1.0
Ok to proceed? (y) y

Creating a new React app in E:\iuh-docker\react-docker-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1324 packages in 18s

268 packages are looking for funding
  run `npm fund` for details

Installing template dependencies using npm...

added 18 packages, and changed 1 package in 2s

268 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...

removed 1 package, and audited 1342 packages in 2s

268 packages are looking for funding
  run `npm fund` for details

8 vulnerabilities (2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

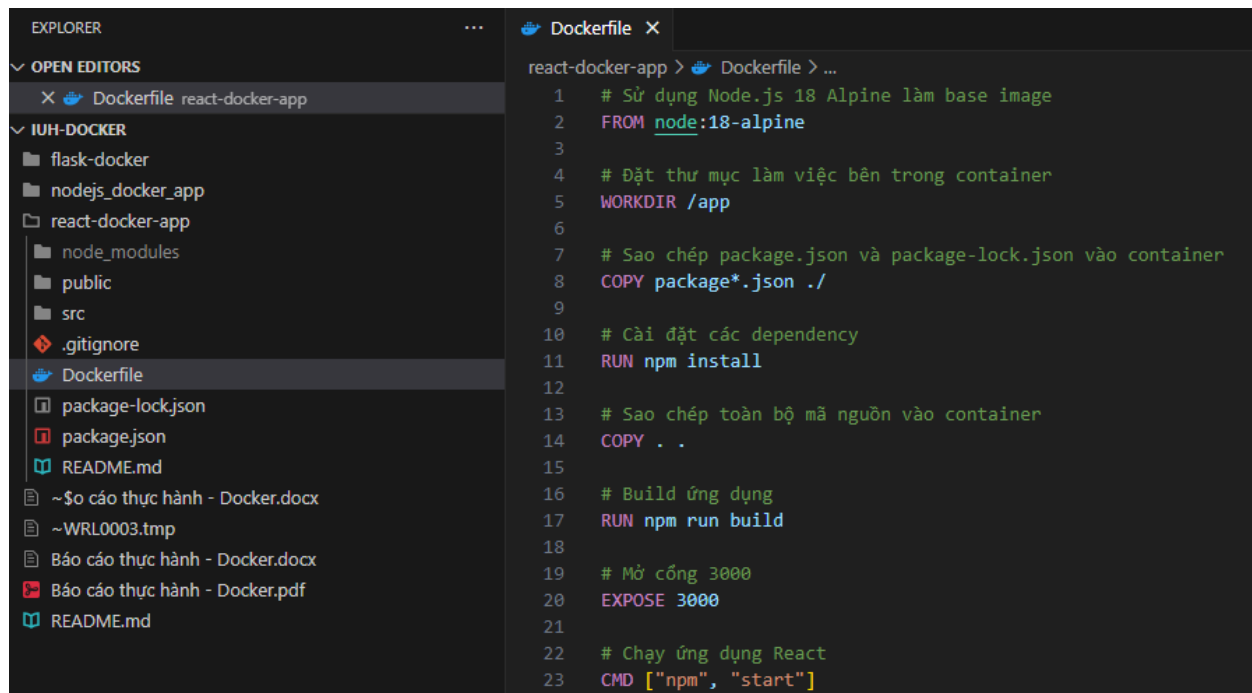
Success! Created react-docker-app at E:\iuh-docker\react-docker-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.
```

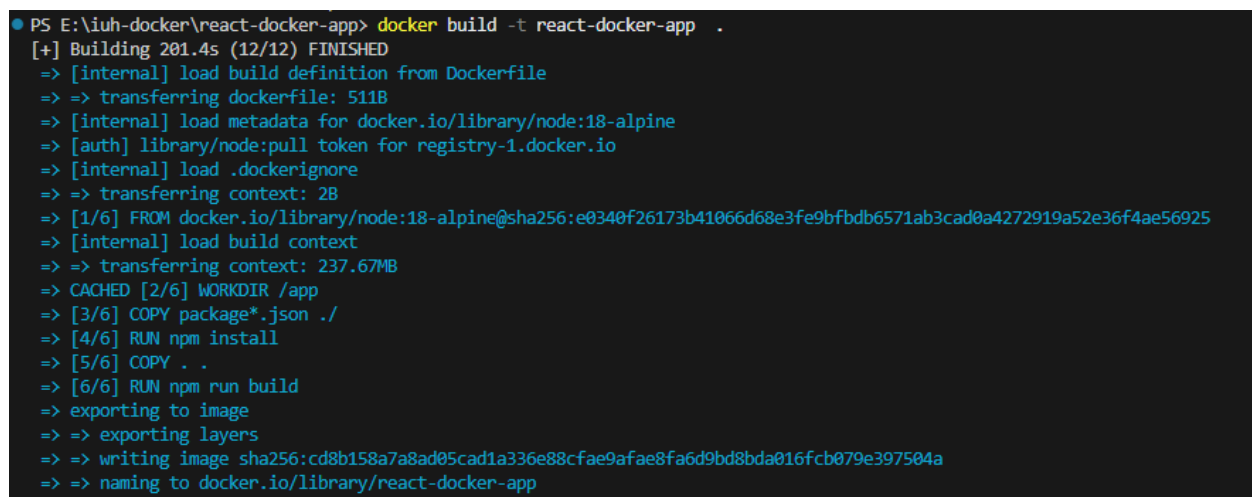
Bước 2: Tạo docker file và cấu hình cho dockerfile



The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the file structure of a project named 'react-docker-app'. The files listed include 'package-lock.json', 'package.json', 'README.md', and several Docker-related files like '~\$o cáo thực hành - Docker.docx'. The 'Dockerfile' is selected and open in the main editor. The Dockerfile content is as follows:

```
1 # Sử dụng Node.js 18 Alpine làm base image
2 FROM node:18-alpine
3
4 # Đặt thư mục làm việc bên trong container
5 WORKDIR /app
6
7 # Sao chép package.json và package-lock.json vào container
8 COPY package*.json ./
9
10 # Cài đặt các dependency
11 RUN npm install
12
13 # Sao chép toàn bộ mã nguồn vào container
14 COPY . .
15
16 # Build ứng dụng
17 RUN npm run build
18
19 # Mở cổng 3000
20 EXPOSE 3000
21
22 # Chạy ứng dụng React
23 CMD ["npm", "start"]
```

Bước 3: Build và run ứng dụng



The screenshot shows a terminal window with the following commands and output:

```
PS E:\iuh-docker\react-docker-app> docker build -t react-docker-app .
[+] Building 201.4s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 511B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/node:18-alpine@sha256:e0340f26173b41066d68e3fe9bfdb6571ab3cad0a4272919a52e36f4ae56925
=> [internal] load build context
=> => transferring context: 237.67MB
=> CACHED [2/6] WORKDIR /app
=> [3/6] COPY package*.json ./
=> [4/6] RUN npm install
=> [5/6] COPY . .
=> [6/6] RUN npm run build
=> exporting to image
=> => exporting layers
=> => writing image sha256:cd8b158a7a8ad05cad1a336e88cfae9afae8fa6d9bd8bda016fcb079e397504a
=> => naming to docker.io/library/react-docker-app
```

```

PS E:\iuh-docker\react-docker-app> docker run -p 3000:3000 react-docker-app

> react-docker-app@0.1.0 start
> react-scripts start

(node:30) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:30) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...

Compiled successfully!

You can now view react-docker-app in the browser.

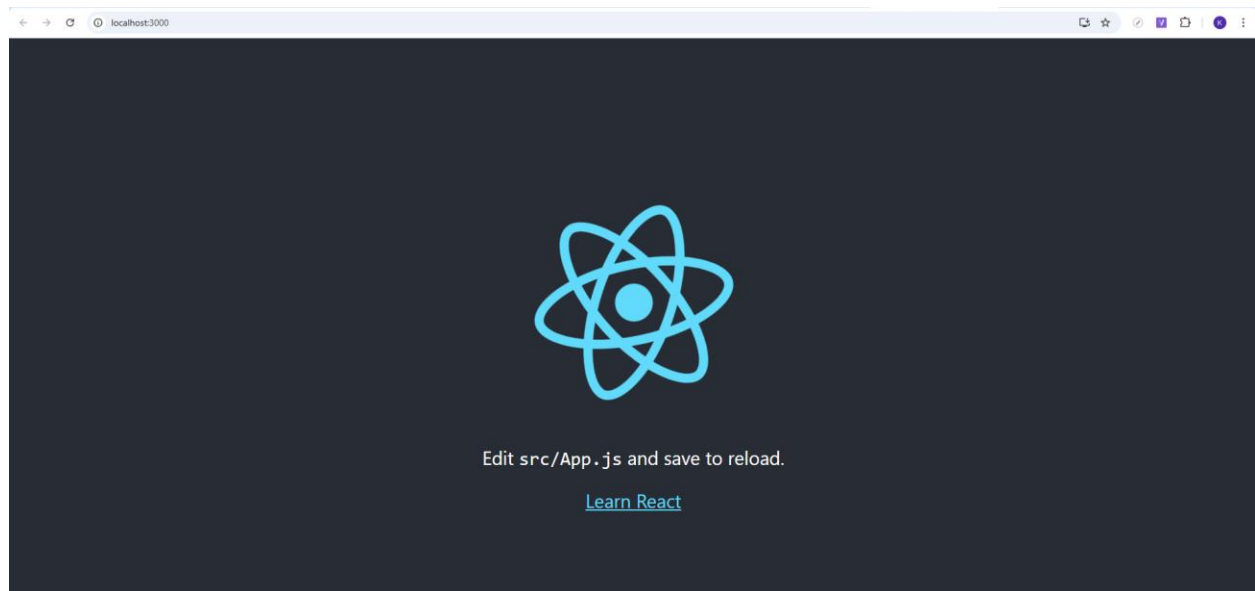
Local:      http://localhost:3000
On Your Network:  http://172.17.0.2:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
Compiling...
Compiled successfully!
webpack compiled successfully

```

Kết quả:



Bài 4: Tạo ứng dụng với Nginx

Bước 1: Tạo thư mục và các file cần thiết

Bước 2: Tạo file index.html và cấu hình docker file

```
index.html U X Dockerfile U
nginx-docker-app > <> index.html > ...
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>My Nginx Page</title>
7 </head>
8 <body>
9     <h1>Hello, this is a static website served by Nginx!</h1>
10 </body>
11 </html>
12
```

```
index.html U Dockerfile U X
nginx-docker-app > Dockerfile > ...
1 # Sử dụng Nginx latest làm base image
2 FROM nginx:latest
3
4 # Sao chép file index.html vào thư mục mặc định của Nginx
5 COPY index.html /usr/share/nginx/html/index.html
6
7 # Mở cổng 80
8 EXPOSE 80
9
10 # Chạy Nginx
11 CMD ["nginx", "-g", "daemon off;"]
12
```

Bước 3: Build ứng dụng và start container.

```
PS E:\iuh-docker\nginx-docker-app> docker build -t nginx-app .
[+] Building 0.3s (7/7) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 310B 0.0s
=> [internal] load metadata for docker.io/library/nginx:latest 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 318B 0.0s
=> CACHED [1/2] FROM docker.io/library/nginx:latest 0.0s
=> [2/2] COPY index.html /usr/share/nginx/html/index.html 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:6775a5f0216fce54cdd4903b36390450a8f8ce0e5868ce1199117f024df9f7f2 0.0s
=> => naming to docker.io/library/nginx-app 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/x7tut3wti4sdx9ujvzxy145
PS E:\iuh-docker\nginx-docker-app>
```

```
PS E:\iuh-docker\nginx-docker-app> docker run -p 8080:80 nginx-app
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/03/22 14:54:41 [notice] 1#1: using the "epoll" event method
2025/03/22 14:54:41 [notice] 1#1: nginx/1.27.4
2025/03/22 14:54:41 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/03/22 14:54:41 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/03/22 14:54:41 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/03/22 14:54:41 [notice] 1#1: start worker processes
2025/03/22 14:54:41 [notice] 1#1: start worker process 29
2025/03/22 14:54:41 [notice] 1#1: start worker process 30
2025/03/22 14:54:41 [notice] 1#1: start worker process 31
2025/03/22 14:54:41 [notice] 1#1: start worker process 32
2025/03/22 14:54:41 [notice] 1#1: start worker process 33
2025/03/22 14:54:41 [notice] 1#1: start worker process 34
2025/03/22 14:54:41 [notice] 1#1: start worker process 35
2025/03/22 14:54:41 [notice] 1#1: start worker process 36
2025/03/22 14:54:41 [notice] 1#1: start worker process 37
2025/03/22 14:54:41 [notice] 1#1: start worker process 38
2025/03/22 14:54:41 [notice] 1#1: start worker process 39
2025/03/22 14:54:41 [notice] 1#1: start worker process 40
2025/03/22 14:54:41 [notice] 1#1: start worker process 41
2025/03/22 14:54:41 [notice] 1#1: start worker process 42
2025/03/22 14:54:41 [notice] 1#1: start worker process 43
2025/03/22 14:54:41 [notice] 1#1: start worker process 44
```

Kết quả:

← → ↺ ⓘ localhost:8080

Hello, this is a static website served by Nginx!

Bài 5: Tạo ứng dụng Go

Bước 1: Tạo file và cấu hình docker file

```
go-docker-app > main.go U X Dockerfile U X
go-docker-app > main.go
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6 )
7
8 func handler(w http.ResponseWriter, r *http.Request) {
9     fmt.Fprintf(w, "Hello, Docker Go!")
10 }
11
12 func main() {
13     http.HandleFunc("/", handler)
14     fmt.Println("Server is running on port 8080...")
15     http.ListenAndServe(":8080", nil)
16 }
```

```
go-docker-app > Dockerfile U X
go-docker-app > Dockerfile > ...
1 # Sử dụng Go làm base image
2 FROM golang:latest
3
4 # Đặt thư mục làm việc trong container
5 WORKDIR /app
6
7 # Sao chép toàn bộ mã nguồn vào container
8 COPY . .
9
10 # Biên dịch ứng dụng Go thành file thực thi
11 RUN go build -o main .
12
13 # Mở cổng 8080
14 EXPOSE 8080
15
16 # Chạy ứng dụng Go
17 CMD ["/main"]
```

Bước 2: Build và start

```

PS E:\iuh-docker\go-docker-app> docker build -t go-app .
[+] Building 7.7s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 381B
=> [internal] load metadata for docker.io/library/golang:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/golang:latest@sha256:52ff1b35ff8de185bf9fd26c70077190cd0bed1e9f16a2d498ce907e5c421268
=> [internal] load build context
=> => transferring context: 123B
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN go build -o main .
=> exporting to image
=> => exporting layers
=> writing image sha256:ddf150e3427ee3ed2d0ea1f1c34209d7c9dda92a89036674f7ec2f9050d5dc01
=> naming to docker.io/library/go-app
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/1z1b4suaxwz1188mvfuh55mof

```

```

PS E:\iuh-docker\go-docker-app> docker run -p 8080:8080 go-app
Server is running on port 8080...

```

Kết quả:

localhost:8080

Hello, Docker Go!

Bài 6: Sử dụng Multi-stage Build trong Dockerfile

Viết Dockerfile để build một ứng dụng Node.js với hai stage:

Stage 1: Dùng node:18 để build code.

Stage 2: Dùng node:18-alpine để chạy ứng dụng đã build.

Bước 1: Khởi tạo ứng dụng ExpressJS

```
PS E:\iuh-docker\nodejs-docker-app> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (nodejs-docker-app)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to E:\iuh-docker\nodejs-docker-app\package.json:

{
  "name": "nodejs-docker-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}

Is this OK? (yes) yes
```

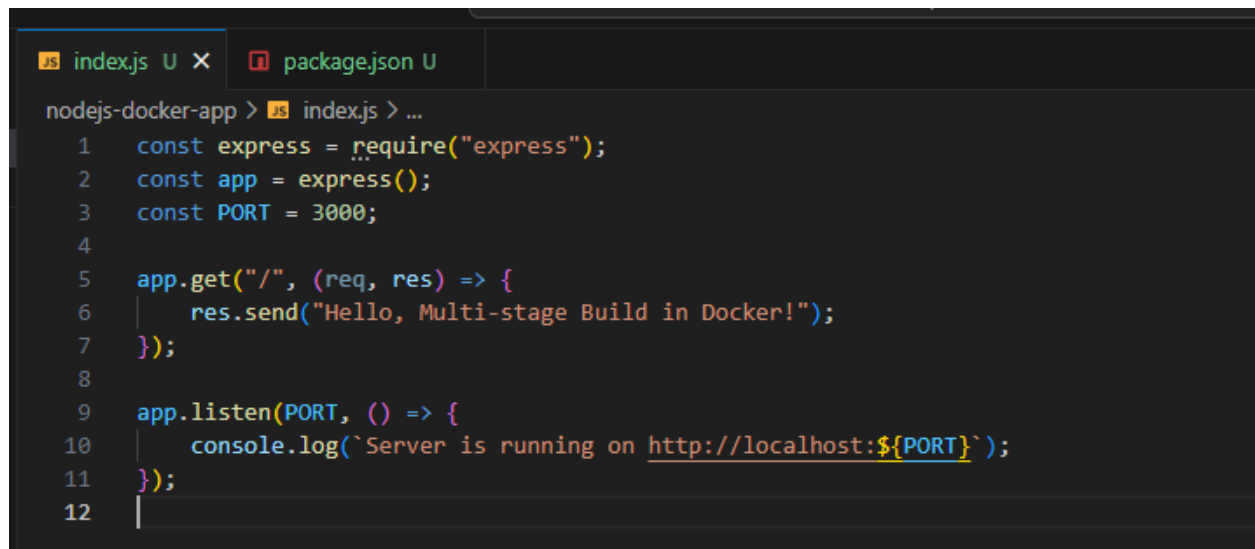
```
PS E:\iuh-docker\nodejs-docker-app> npm install express

added 69 packages, and audited 70 packages in 3m

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

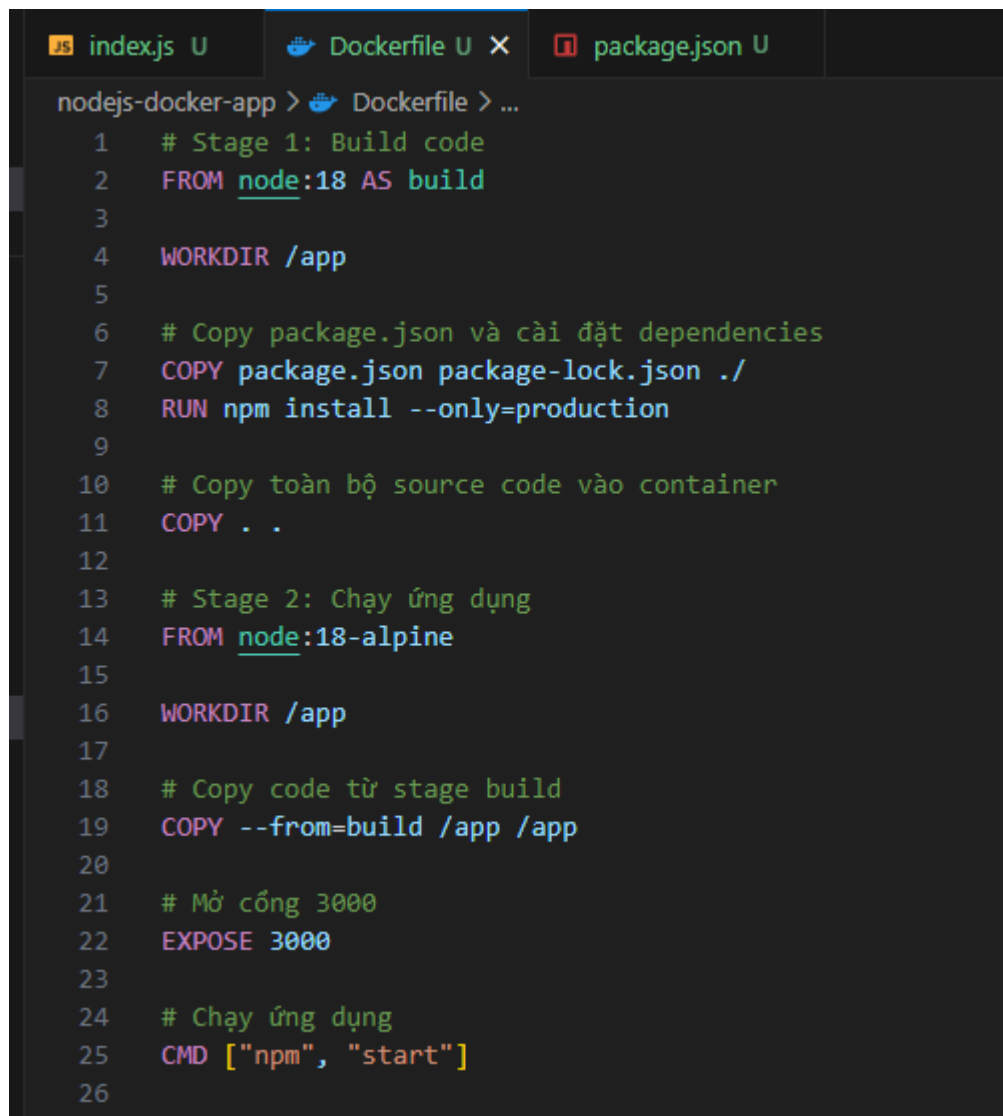
Bước 2: Viết nội dung cho index.js



The screenshot shows a code editor with two tabs: 'index.js' and 'package.json'. The 'index.js' tab is active, displaying the following JavaScript code:

```
nodejs-docker-app > index.js > ...
1  const express = require("express");
2  const app = express();
3  const PORT = 3000;
4
5  app.get("/", (req, res) => {
6    res.send("Hello, Multi-stage Build in Docker!");
7  });
8
9  app.listen(PORT, () => {
10    console.log(`Server is running on http://localhost:${PORT}`);
11  });
12 |
```

Bước 3: Viết Dockerfile với Multi-stage Build



```
nodejs-docker-app > Dockerfile > ...
1  # Stage 1: Build code
2  FROM node:18 AS build
3
4  WORKDIR /app
5
6  # Copy package.json và cài đặt dependencies
7  COPY package.json package-lock.json ./
8  RUN npm install --only=production
9
10 # Copy toàn bộ source code vào container
11 COPY . .
12
13 # Stage 2: Chạy ứng dụng
14 FROM node:18-alpine
15
16 WORKDIR /app
17
18 # Copy code từ stage build
19 COPY --from=build /app /app
20
21 # Mở cổng 3000
22 EXPOSE 3000
23
24 # Chạy ứng dụng
25 CMD ["npm", "start"]
26
```

Bước 4: Build và Start

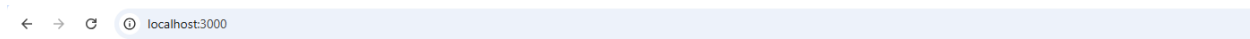
docker build -t multi-node-app .

```
PS E:\iuh-docker\nodejs-docker-app> docker build -t multi-node-app .
[+] Building 8.7s (15/15) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 504B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [internal] load metadata for docker.io/library/node:18
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [build 1/5] FROM docker.io/library/node:18@sha256:7f6bcd8e08a1f81bfb29f5948de5c5507624788db50cadb94ddd439426b7c4
=> => resolve docker.io/library/node:18@sha256:7f6bcd8e08a1f81bfb29f5948de5c5507624788db50cadb94ddd439426b7c4
=> [internal] load build context
=> => transferring context: 2.32MB
=> [stage-1 1/3] FROM docker.io/library/node:18-alpine@sha256:e0340f26173b41066d68e3fe9bfbdb6571ab3cad0a4272919a52e36f4ae56925
=> CACHED [stage-1 2/3] WORKDIR /app
=> CACHED [build 2/5] WORKDIR /app
=> [build 3/5] COPY package.json package-lock.json ./
=> [build 4/5] RUN npm install --only-production
=> [build 5/5] COPY . .
=> [stage-1 3/3] COPY --from=build /app /app
=> exporting to image
=> => exporting layers
=> => writing image sha256:a6c068416e5d00a181d3a38671e0bc4e92d72b6096ed5c2efa07e2af7819a8e5
=> => naming to docker.io/library/multi-node-app
View build details: docker-desktop:///dashboard/build/desktop-linux/desktop-linux/kyt2dp14efczxhgtzef6huld3
```

docker run -p 3000:3000 multi-node-app

```
PS E:\iuh-docker\nodejs-docker-app> docker run -p 3000:3000 multi-node-app
nodejs-docker-app@1.0.0 start
node index.js
Server is running on http://localhost:3000
```

Kết quả:



Hello, Multi-stage Build in Docker!

Bài 7: Sử dụng biến môi trường trong Dockerfile

Bước 1: Tạo ứng dụng python

```
app.py U X Dockerfile U
python-env-docker > app.py
1 import os
2
3 # Đọc biến môi trường APP_ENV, mặc định là 'production' nếu không được đặt
4 app_env = os.getenv("APP_ENV", "production")
5
6 print(f"Ứng dụng đang chạy trong môi trường: {app_env}")
7
```

```
python-env-docker > Dockerfile U X
python-env-docker > ...
1 # Sử dụng Python 3.9 làm base image
2 FROM python:3.9
3
4 # Đặt thư mục làm việc trong container
5 WORKDIR /app
6
7 # Sao chép mã nguồn vào container
8 COPY . .
9
10 # Thiết lập biến môi trường
11 ENV APP_ENV=development
12 |
13 # Chạy ứng dụng
14 CMD ["python", "app.py"]
15
```

Bước 2: Build và Start

```
PS E:\iuh-docker\python-env-docker> docker build -t python-env-app .
• [+] Building 2.3s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 332B
=> [internal] load metadata for docker.io/library/python:3.9
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/library/python:3.9@sha256:bc2e05bca883473050fc3b7c134c28ab822be73126ba1ce29517d9e8b7f3703b
=> [internal] load build context
=> => transferring context: 600B
=> CACHED [2/3] WORKDIR /app
=> [3/3] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:85201606cfefb960cb29bb9b8e8eb5e1fcf76cfa18fdff3be7ab8d4f7550fbf4
=> => naming to docker.io/library/python-env-app

• PS E:\iuh-docker\python-env-docker> docker run python-env-app
Ứng dụng đang chạy trong môi trường: development
```