

# TÀI LIỆU KHÓA HỌC “DEPLOY REACT/NODE.JS”

Tác giả: Hỏi Dân IT & Eric

Version: 4.0

Note thay đổi:

- Cập nhật Build Docker cho Next.JS
- Bổ sung thêm video từ #80 tới #84

<b>Chapter 0: Giới Thiệu Về Khóa Học</b>	<b>5</b>
#0.1. Demo Kết Quả Đạt Được Khi Kết Thúc Khóa Học	5
#0.2. Yêu Cầu Của Khóa Học	6
#0.3. Về Khóa Học này	7
#0.4. Về Tác Giả	9
<b>Chapter 1: Client Apps (Browser)</b>	<b>10</b>
#1. Browser and Javascript - Phương thức truyền thống	10
#2. Browser Type/Version	11
#3. Assets Files: Typescript (JS)/Css/Image	12
#4. Các Vấn Đề Tồn Động	12
#5. Why Webpack ?	13
<b>Chapter 2: Deploy React App</b>	<b>14</b>
#6. Create React App - Hello World	14
#7. Babel vs Webpack vs React Scripts	15
#8. Build Command (CRA)	16
#9. Run Build React Localhost (CRA)	17
#10. Cross Platform với cross-env	18
#11. Test Localhost và Test Build (cross env)	18
#12. Deploy Github Page (CRA)	19
#13. Deploy with Redux/React router (CRA)	20
#14. React Vite - Hello world	22
#15. React Vite Build Local	23
#16. Deploy Github Page (Vite)	24
#17. Deploy with Vercel	25
#18. Nhận Xét Về Deploy Frontend	26
<b>Chapter 3: Deploy Node.JS Server</b>	<b>27</b>
#19. Build Node.JS Server	27
#20. Phân Biệt Development và Production	28
#21. Nodemon, Babel & devDependencies	29
#22. Babel Run Production	30
#23. Run forever với PM2 (Auto Restart)	32
#24. Logging	33
#25. From Heroku to Render (Node.JS)	35
#26. Deploy with Render (Only Backend)	36
#27. Create SQL Database (with supabase)	36
#28. Deploy Backend và SQL Database	36
#29. Nosql Database với MongoDB Atlas	37
#30. Deploy Backend và MongoDB	37
#31. Nhận Xét Về Deploy Backend	38
<b>Chapter 4: Docker và Nginx</b>	<b>39</b>

#32. Nginx Introduction	39
#33. Nginx với Docker	40
#34. Nginx Basic Command	42
#35. Nginx Default Folder	43
#36. Nginx với Docker Compose	44
#37. Nginx Customize HTML	45
#38. Run React App With Nginx (Basic)	46
<b>Chapter 5: Nginx Web Server</b>	<b>47</b>
#39. Nginx Config Files	47
#40. Syntax Config Files	48
#41. Setting Up Virtual Servers	50
#42. MIME Types	51
#43. Handling Errors	51
#44. Run React App With Nginx (Advance)	52
#45. Giới thiệu Serving Static Content	53
#46. Giới thiệu Compression và Decompression	53
#47. Nhận xét về web server (only Frontend)	53
<b>Chapter 6: Nginx Reverse Proxy &amp; Load Balancing</b>	<b>54</b>
#48. Phân biệt Proxy và Firewall	54
#49. Phân biệt Reverse Proxy và Forward Proxy	54
#50. Ví dụ App React Với Proxy	55
#51. Build React App với Nginx và Backend	55
#52. Nginx Proxy Pass	55
#53. Load Balancing: upstream	56
#54. Setup Docker Compose Backend và Database Service	56
#55. Setup Docker Compose React và Nginx	56
#56. Run Full App với Docker	56
#57. Bỏ trợ - Docker Compose Dạng Basic (Mapping port)	56
<b>Chapter 7: Run Web App với Hosting/Domain</b>	<b>57</b>
#58. Hosting Server Website	57
#59. Hướng Dẫn Mua Hosting	59
#60. Công cụ kết nối vào Hosting	61
#61. Sử Dụng FileZilla (MacOS)	62
#61.1 Cấu Hình Hosting (Basic)	63
#61.2. Setup Docker với Hosting	65
#62. Run app with Hosting	67
#63. Check IP Client (Docker Logs)	68
#64. Tên miền Website (Domain)	69
#65. Mua Tên Miền	70
#66. Điều Cần Làm Sau Khi Mua Domain	71
#67. Mapping Domain tới Hosting	72

#68. Mapping Domain tới Github Page	73
<b>Chapter 8 : Securing The Apps</b>	<b>74</b>
#69. Setup Run Docker (without sudo)	74
#70. SSL Certificate	76
#71. Let's Encrypt và Certbot	78
#72. Run Nginx với SSL (Basic)	79
#73. Run Nginx full App với SSL (Staging)	80
#74. Check Renew Certificate	82
#75. Run Nginx full App với SSL (Production)	82
#76. Các bước để tự chạy ứng dụng của bạn	83
#77. Nhận xét về Build & Deploy Website từ A tới Z	84
<b>Chapter 9: Run Web App Free với Virtual Host (Localhost)</b>	<b>85</b>
#78. Setup Docker với Virtual Box & Linux Alpine	85
#79. Copy Files Từ Windows Host Vào Virtual Machine Linux	96
<b>Chapter 10: Build App với Next.JS và Nginx</b>	<b>99</b>
#80. Deploy Hosting FREE cho Next.JS	99
#81. Build Next.JS với Docker	100
#82. Lưu Ý Khi Build Docker với Next.JS	101
#83. Build NextJS với Docker Compose	102
#84. Build NexJS với Nginx	102
<b>Lời Kết</b>	<b>103</b>

## Chapter 0: Giới Thiệu Về Khóa Học

### #0.1. Demo Kết Quả Đạt Được Khi Kết Thúc Khóa Học

Video demo (coming soon).

#### Một vài highlight của khóa học:

- Hiểu rõ như thế nào là **'build' source code cho production**. Lâu nay, việc chúng ta code và chạy localhost, là chế độ 'development'. Còn khi triển khai và chạy thực tế, chúng ta cần build ở chế độ 'production'
- Triển khai ứng dụng Frontend (only FE React) với tên miền của Github (Github Pages) và tên miền của Vercel. **Cách triển khai này hoàn toàn miễn phí.**
- Triển khai ứng dụng Backend (only BE Node.JS) với tên miền của Render. Ngoài ra, triển khai database SQL (Postgres) và database NoSQL (MongoDB). **Cách triển khai này cũng miễn phí.**
- **Sử dụng Nginx web server** với Docker (chạy ứng dụng website tại localhost với Nginx)
- **Nginx + Docker + VPS : chạy website thực tế với Hosting và Domain** ( SSL miễn phí với Let's Encrypt) thông qua Docker

## #0.2. Yêu Cầu Của Khóa Học

Các yêu cầu cần biết trước khi thực hành khóa học:

Do khóa học này là khóa học triển khai ứng dụng lên production (môi trường chạy thực tế), thành ra, các kiến thức về chạy source code mình không hướng dẫn. Vì đơn giản, bạn phải code ra được website, thì mới nghĩ tới việc triển khai nó.

Một vài yêu cầu kiên quyết:

- **Cài đặt môi trường Node.JS** : trong khóa học này, máy tính của mình dùng **version Node.JS 14.17.0**  
Để hạn chế lỗi, các bạn nên dùng version trên. Có thể dùng NVM để sử dụng nhiều version của Node.JS trên cùng máy tính  
Về NPM, xem tại:  
[https://www.youtube.com/watch?v=ccjKHLyo4IM&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC\\_9Vql&index=40](https://www.youtube.com/watch?v=ccjKHLyo4IM&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC_9Vql&index=40)
- **Cài đặt Git**: các thao tác với Git và sử dụng Github, mình không hướng dẫn. Bạn nào chưa biết dùng Git, xem tại đây:  
<https://www.youtube.com/watch?v=-BtolPy15fg&list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo>
- **Cài đặt Docker Desktop**: tất cả công cụ cần thiết, mình sẽ cài đặt thông qua Docker. Bạn nào chưa biết dùng Docker, thì xem tại đây:  
<https://www.youtube.com/watch?v=tZhQPlasDLY&list=PLncHg6Kn2JT4EcU8eTdmojGO1StFDILe2&index=1>
- **Cài đặt Visual Studio Code**: đây là công cụ dùng để code

### #0.3. Về Khóa Học này

#### 1. Tại sao mình làm khóa học này?

Ban đầu, **mục đích khóa học Udemy của mình, là phục vụ kiến thức cho các bạn học để đi làm cho công ty CNTT.**

Tuy nhiên, lại có rất nhiều học viên Udemy của Hỏi Dân IT, sau khi học xong các khóa học của mình, mong muốn **triển khai nó lên “thực tế”**.

Thực tế ở đây, là một website chạy thực sự, chứ không phải website demo nữa (phục vụ mục đích quảng bá thương hiệu bản thân và mô hình kinh doanh nhỏ).

Đa phần những học viên này, là các bạn tự học IT, không có nguyện vọng làm cho công ty CNTT nên có nhu cầu như vậy.

Vì vậy, mình làm khóa học này, là để giải quyết vấn đề trên.

#### 2. Chuyện VPS (Hosting)

**Special thanks to bạn Kiên Quốc đã donate VPS cho mình thực hiện khóa học này.**

#### 3. SOS thì làm sao

Với các bạn học viên Udemy, để đảm bảo quyền lợi, các bạn chủ động inbox qua fanpage Hỏi Dân IT để nhận được sự hỗ trợ trong quá trình học tập nhé:

<https://www.facebook.com/askITwithERIC/>

#### **4. Về chuyện leak khóa học và mua lậu**

Mình biết rất nhiều bạn khi học khóa học này của mình, là mua lậu qua bên thứ 3. chuyện này là hoàn toàn bình thường, vì thương hiệu “Hoi Dan IT” đang ngày càng khẳng định được vị thế của mình.

Nhiều bạn hỏi mình, sao mình không ‘chặn việc mua lậu’. nói thật, nếu mình làm, là làm được đấy, cơ mà nó sẽ gây ra sự bất tiện cho học viên chân chính (con sâu làm rầu nồi canh). Với lại, ngay cả hệ điều hành windows, còn bị crack nữa là @@

Mình cũng có 1 bài post facebook về chuyện này:

<https://www.facebook.com/askitwitheric/posts/pfbid02gyasktd3semqxt6nevnvwh4c8epzu3i7kpzhr7s7gmmfcvucyz96eb8avnvgnhl>

Với các bạn học viên chân chính, mình tin rằng, những cái các bạn nhận được từ mình khi đã chấp nhận đầu tư, nó sẽ hoàn toàn xứng đáng. vì đơn giản, với cá nhân mình, khách hàng là thượng đế.

VỚI CÁC BẠN MUA LẬU, MÌNH CHỈ MUỐN CHIA SẺ THẾ NÀY:

**1. TRÊN ĐỜI NÀY, CHẴNG CÓ GÌ CHẤT LƯỢNG MÀ MIỄN PHÍ CẢ.**

VIỆC BẠN MUA LẬU QUA BÊN THỨ 3, LÀ GIÚP BỌN CHÚNG LÀM GIÀU VÀ GÂY THIẾT HẠI CHO TÁC GIẢ.

NẾU NHÌN VỀ TƯƠNG LAI => Càng ngày càng ít tác giả làm khóa học => NGƯỜI BỊ HẠI CUỐI CÙNG VẪN LÀ HỌC VIÊN

**2. HÃY HỌC THÓI QUEN TRÂN TRỌNG GIÁ TRỊ LAO ĐỘNG**

NÓ LÀ THÓI QUEN, CŨNG NHƯ SẼ LÀ MỘT PHẦN TÍNH CÁCH CỦA BẠN.

ĐỪNG VÌ NGHÈO QUÁ MÀ LÀM MẤT ĐI TÍNH CÁCH CỦA BẢN THÂN.

NẾU KHÓ KHĂN, CỨ INBOX MÌNH, MÌNH HỖ TRỢ. VIỆC GÌ PHẢI LÀM VẬY =))

**3. MÌNH ĐÃ TỪNG LÀ SINH VIÊN GIỐNG BẠN, MÌNH HIỂU TẠI SAO CÁC BẠN LÀM VẬY. HÃY BIẾT CHO ĐI. SỐNG ÍCH KỶ, THÌ THEO LUẬT NHÂN QUẢ ĐẤY, CHẴNG CÓ GÌ LÀ NGẪU NHIÊN CẢ**

**4. NẾU BẠN THẤY KHÓA HỌC HAY, HÃY BIẾT DONATE ĐỂ ỦNG HỘ TÁC GIẢ. LINK DONATE: <https://hoidanit.github.io/official/donate>**

Hành động nhỏ nhưng mang ý nghĩa lớn. Hãy vì 1 cộng đồng IT Việt Nam phát triển. Nếu làm như các bạn, có lẽ chúng ta đã không có Iphone, không có Apple như ngày nay rồi @@



#### **#0.4. Về Tác Giả**

Mọi thông tin về Tác giả Hỏi Dân IT, các bạn có thể tìm kiếm tại đây:

Youtube “Hỏi Dân IT” : <https://www.youtube.com/@hoidanit>

Tiktok “Hỏi Dân IT” : <https://www.tiktok.com/@hoidanit>

Fanpage “Hỏi Dân IT” : <https://www.facebook.com/askITwithERIC/>

Website chính thức: <https://hoidanit.com.vn/>

Udemy Hỏi Dân IT: <https://www.udemy.com/user/eric-7039/>

## Chapter 1: Client Apps (Browser)

### #1. Browser and Javascript - Phương thức truyền thống

#### 1. Template mẫu

Tài liệu: <https://themewagon.com/theme-tag/portfolio-template/>

#### Ví dụ demo:

<https://themewagon.com/themes/free-bootstrap-4-html5-portfolio-website-template-videograph/>

<https://themewagon.github.io/videograph/index.html>

=> <https://github.com/themewagon/videograph>

#### 2. Cách làm

- Nguyên tắc: **Browser chỉ có thể hiểu được Javascript và CSS.**

- Khi đọc file HTML, để hiển thị Javascript và CSS, bắt buộc, phải 'import' và sử dụng.

Cụ thể:

+ **CSS được import thông qua tag <link rel="stylesheet" /> => đặt ở đầu file**

+ **Javascript được import thông qua tag <script></script> => đặt trước body tag**

<https://stackoverflow.com/questions/30653081/why-scripts-at-the-end-of-body-tag>

- Với cách làm ở trên, chúng ta đang :

+ Những file html nào cần sử dụng Javascript => import vào

+ Nếu không muốn import nhiều file, tạo 1 file lớn (đã bao gồm file con) => import

---

- Thực tế, để phát triển 1 website, cách làm trên không hiệu quả :v

## #2. Browser Type/Version

### 1. Các loại web browser

[https://en.wikipedia.org/wiki/List\\_of\\_web\\_browsers](https://en.wikipedia.org/wiki/List_of_web_browsers)

Phân thành 2 loại chính:

- Chạy trên PC (personal computer)
- Chạy trên mobile

=> có rất nhiều loại browsers, câu hỏi đặt ra, là làm sao ứng dụng web có thể chạy tốt trên tất cả loại trên ?

### 2. Lịch sử phát triển của Javascript

[https://www.w3schools.com/js/js\\_history.asp](https://www.w3schools.com/js/js_history.asp)

[https://www.w3schools.com/js/js\\_versions.asp](https://www.w3schools.com/js/js_versions.asp)

- Tất cả các trình duyệt web hiện đại (Google Chrome/FireFox...) đều support version ES6
- Tuy nhiên, để đảm bảo ứng dụng chạy tốt trên tất cả browser, thông thường, ứng dụng web sẽ được 'biên dịch' ở dạng ES5

**Tức là, cần có trình dịch code, giúp chuyển hóa cú pháp ES6 (hoặc mới hơn) về code ES5.**

Ví dụ:

ES6: `const name = 'eric';`

=> ES5: `var name = 'eric';`

### #3. Assets Files: Typescript (JS)/Css/Image

#### 1. Typescript

Browser không thể hiểu TS, chỉ có thể hiểu JS và CSS

**=> cần dịch Typescript => Javascript (ES5)**

#### 2. CSS

- Nếu sử dụng các công cụ viết CSS, như SASS, browser không thể đọc được.

**=> cần dịch về CSS**

#### 3. Images

- Đôi khi cần tối ưu hóa quá trình load ảnh (chuyển trang qua lại, refresh trang)

**=> cache image**

### #4. Các Vấn Đề Tồn Động

#### 1. Theo cách truyền thống

- import javascript vào từng file

+ khó có thể mở rộng (scale), vì nếu có quá nhiều scripts, và không load được scripts  
**=> lỗi**

- sử dụng 1 file duy nhất => import tất cả vào

+ tính mở rộng (khi khai báo biến, kích thước file...)  
+ khó maintain

#### 2. Cách hiện tại

- Làm sao để dịch code:

+ dịch từ TS sang JS  
+ dịch từ Sass sang css  
+ tối ưu hóa Images...

- Đảm bảo code chạy tốt trên tất cả trình duyệt

- Với developer, sử dụng được các công cụ (functions, syntax) đã hỗ trợ của Javascript

## #5. Why Webpack ?

Tài liệu: <https://webpack.js.org/concepts/why-webpack/>

### 1. Webpack

- Bundle your scripts/styles/assets
- Hiểu 1 cách đơn giản, cách webpack hoạt động:

input: project code

+ html

+ javascript (typescript)

+ CSS (sass)

+ images

output: bundle

+ Tối ưu hóa js/css/images...

### 2. Ví dụ khi sử dụng webpack

todo...

## Chapter 2: Deploy React App

### #6. Create React App - Hello World

Tài liệu:

<https://create-react-app.dev/docs/getting-started>

#### 1. Create project

Sử dụng câu lệnh:

**npx create-react-app my-app**

You'll need to have Node  $\geq 14$  on your local development machine

(tương lai sẽ yêu cầu version lớn hơn)

---

Link download:

<https://drive.google.com/file/d/18saqkD81UDYDAWoFW32wbJpTNpT3cZoS/view?usp=sharing>

#### 2. Run local

Sử dụng câu lệnh:

**npm start**

- Các công việc Create-react-app đã làm:

- + Tích hợp sẵn webpack => tối ưu hóa việc chạy dự án (từ size, performance...)
- + Developer có thể sử dụng syntax mới nhất của Javascript
- + Hot reloading

## #7. Babel vs Webpack vs React Scripts

<https://stackoverflow.com/questions/46337918/difference-between-webpack-babel-and-react-scripts>

Tóm gọn tác dụng của 3 thư viện trên:

**Babel:** Trình biên dịch code javascript. Khi sử dụng babel, chúng ta có thể sử dụng bất kỳ version nào của Javascript mà không lo bị lỗi khi chạy với browser/server.

Vì đơn giản, Babel sẽ dịch code javascript về version cũ hơn (thông thường là ES5)

**Webpack:** giúp tối ưu hóa hiệu năng (performance) của ứng dụng. Hiểu đơn giản webpack tương tự như winrar, nó sẽ 'nén' source code lại => giảm size bundle => website chạy nhanh hơn

**React Script:** là thư viện dùng để tạo nên 1 dự án React với react-create-app tool. Khi sử dụng công cụ này, nó đã tích hợp sẵn Babel và Webpack, như vậy developer chỉ lo code mà không cần quan tâm tới việc cấu hình Babel/Webpack

## #8. Build Command (CRA)

### 1. npm run build

"build": "react-scripts build",

- Khi chạy câu lệnh "npm run build", câu lệnh "react-scripts build" sẽ được thực thi

- Câu lệnh trên sẽ làm các nhiệm vụ:

+ Sử dụng webpack/babel... chuyển hóa code JS về ES5

+ Tối ưu hóa files (bundle)

+ Tất cả file (CSS/JS/Node\_modules) => bundle

Lưu ý: khi chạy câu lệnh npm run build, đồng nghĩa với :

NODE\_ENV=production

khi chạy câu lệnh npm start, NODE\_ENV=development

### 2. Phân biệt environment

- Chạy chế độ build => sẵn sàng để triển khai lên production

+ dung lượng app (size nhỏ => load nhanh)

+ tối ưu hóa code (performance)

+ chỉ cài đặt (chuyển hóa) các package (library) không phải là devDependencies

- Chạy chế độ local => development => code chạy được là được

+ chạy tất cả package

### 3. Kết quả build

<https://stackoverflow.com/questions/41495658/use-custom-build-output-folder-when-using-create-react-app>



## #9. Run Build React Localhost (CRA)

### 1. Run at localhost

- Mặc định, chỉ cần mở file index.html thì run được, but ???
- Lý do lỗi: không tìm thấy file

### 2. Homepage

<https://stackoverflow.com/questions/44371052/why-wont-react-production-build-run-on-the-browser>

- update file package.json với homepage  
"homepage": "."

- Nhược điểm:
  - + Chạy local: không cần homepage
  - + Chạy build (production): cần homepage

## #10. Cross Platform với cross-env

<https://stackoverflow.com/questions/43011207/using-homepage-in-package-json-without-messing-up-paths-for-localhost>

<https://www.npmjs.com/package/cross-env>

### 1. Cross-env

- Mục đích: không phân biệt platform, các câu lệnh đều chạy được

<https://www.npmjs.com/package/cross-env>

**Cài đặt: `npm install --save-exact cross-env@7.0.3`**

## #11. Test Localhost và Test Build (cross env)

Remove homepage

Update build command:

**Ứng với "homepage": " ." //dấu chấm**

"build": "cross-env PUBLIC\_URL=./ react-scripts build"

**Ứng với "homepage": " /eric"**

"build": "cross-env PUBLIC\_URL=/eric react-scripts build"

- chạy local ok: `npm start`

- chạy build ok: `npm run build`

## **#12. Deploy Github Page (CRA)**

### **1. Chuẩn bị**

- Github account
- Thao tác cơ bản với github

### **2. Github Page**

- Deploy simple HTML page

### **3. Deploy project**

- update home page => build command
- create repo (ứng với homepage)
- git init...
- push master...

github => settings => pages

--save (wait for a moment)

### #13. Deploy with Redux/React router (CRA)

Tài liệu:

<https://create-react-app.dev/docs/deployment/#notes-on-client-side-routing>

<https://github.com/rafgraph/spa-github-pages>

Download projects:

<https://drive.google.com/drive/folders/19oVW8mglenpE7EPCxLWoikKVsh3sfNOT?usp=sharing>

- react 18
- react router 6
- redux toolkit
- sass

Chia 2 folder (build/source code) => build => overwrite .git

Note: cần test local trước khi deploy

update 2 files, index.html + 404.html

```
var pathSegmentsToKeep = 1;
```

**+ update file index.html -> inside public folder**

**+ update react router basename**

**+ add 404.html**

**todo:**

- đưa basename = .env

```
const router = createBrowserRouter(  
  [  
    {  
      path: "/",  
      element: <App />,  
      errorElement: <ErrorPage />,  
      children: [  
        { index: true, element: <Home /> },  
  
        {  
          path: "about",  
          element: <About />,  
        },  
        {  
          path: "contact",  
          element: <Contact />,  
        },  
      ],  
    },  
  ],  
  {  
    basename: '/react-router-redux'  
  });
```

## #14. React Vite - Hello world

### Tài liệu:

Download Project:

<https://drive.google.com/file/d/1-WdO9Y6mlqkw3R7W5jObn7Y541kw7ISb/view?usp=sharing>

### 1.React Vite

- Về Vite: <https://vitejs.dev/guide/>
- Vite là công cụ có mục đích 'giống webpack', dùng để build source code
- Vite ra đời với mục đích giúp khắc phục nhược điểm của webpack:  
<https://vitejs.dev/guide/why.html>

Trong khóa học này, mình sử dụng Vite version 2.x để demo (chạy được với node.js v14.17.0)

Bạn nào dùng Vite version cao hơn thì cài Node.JS version cao hơn nhé.

### 2.Run Project

Sau khi download dự án về, các công việc cần làm:

- Cài đặt thư viện cần thiết với câu lệnh: **npm i**
- Chạy project với câu lệnh: **npm start** hoặc **npm run dev**

Với version mới của Vite (4.x), project sẽ chạy trên port 5473, tuy nhiên, với Vite (2.x), chạy trên port 3000.

Tham số port này thay đổi được, và mình sẽ hướng dẫn ở các video tiếp theo :v

## #15. React Vite Build Local

### 1. Chuẩn bị môi trường production

**Change default port:**

<https://stackoverflow.com/questions/70446474/how-to-set-vite-preview-production-port>

**Sử dụng .env:**

<https://stackoverflow.com/questions/70883903/loading-env-variables-in-react-app-using-vite>

**Sử dụng .env với file config:**

<https://stackoverflow.com/questions/66389043/how-can-i-use-vite-env-variables-in-vite-config-js>

**Change basename với .env**

//todo

**env.production**

<https://vitejs.dev/guide/env-and-mode.html>

### 2. Build command

- Npm run build
- Npm run preview  
Chỉ chạy câu lệnh preview, sau khi đã build rồi (vì không build, thì lấy đâu ra preview)

Mục đích của câu lệnh preview, là giúp xem nhanh project ở chế độ build.

## #16. Deploy Github Page (Vite)

### 1. Chuẩn bị

- Setup basename:
  - + Set up router basename
  - + Setup project basename :  
<https://vitejs.dev/guide/static-deploy.html#github-pages>
- Setup file index.html, 404.html : <https://github.com/rafgraph/spa-github-pages>

### 2. Deploy

Tài liệu: <https://vitejs.dev/guide/static-deploy.html>

#### File vite.config.js

```
import { defineConfig, loadEnv } from 'vite';
import react from '@vitejs/plugin-react'

export default ({ mode }) => {
  process.env = { ...process.env, ...loadEnv(mode, process.cwd()) };
  // import.meta.env.VITE_NAME available here with: process.env.VITE_NAME
  // import.meta.env.VITE_PORT available here with: process.env.VITE_PORT

  return defineConfig({
    plugins: [react()],
    base: process.env.VITE_BASE_URL,
    server: {
      port: process.env.VITE_PORT,
    },
  });
}
```



## #17. Deploy with Vercel

### 1. Chuẩn bị

Đã đẩy source code dự án lên Github

**Lưu ý: không cần thiết phải setup basename khi deploy với Vercel**

Setup 2 repos: 1 CRA (create react app) và 1 Vite

### 2. Thực hành

Các bước thực hiện:

- Tạo 1 tài khoản Vercel: <https://vercel.com/>
- Tạo project với Vercel, kết nối project với Github
- Setup build

**Đối với Vite, cần thêm file vercel.json ở thư mục root**

<https://stackoverflow.com/questions/70870735/react-js-vite-application-returns-404-when-refreshed-on-route>

```
{
  "rewrites": [
    {
      "source": "/(.*)",
      "destination": "/"
    }
  ]
}
```

## **#18. Nhận Xét Về Deploy Frontend**

- Chỉ deploy static page: HTML/CSS/Javascript (cần build source code nếu cần thiết)
- **Không có backend**
- **Không có database**

## Chapter 3: Deploy Node.JS Server

### #19. Build Node.JS Server

#### 1. Build server khác gì so với build app browsers ?

##### Với build app cho Browsers (Frontend):

- Cần đảm bảo chạy tốt trên nhiều loại browsers => **dịch code về ES5**
- Cần tối ưu hóa size code -> hạn chế networking loading + tốt cho performance
- Browsers là công cụ sử dụng phía users (client - người dùng hệ thống).  
RAM/CPU... là phụ thuộc vào users (người ít, người nhiều ...) => giệt lags...

##### Với build app cho Server (Backend):

- Nếu code đã chạy local (tại máy tính bạn), thì server chỉ cần có môi trường giống máy tính bạn, là code auto chạy tốt.
- Không nhất thiết phải dịch code JS về ES5. Chỉ cần dịch code khi:
  - + Sử dụng Typescript (cần dịch về javascript)
  - + Version Node.JS không hỗ trợ (ví dụ, bạn dùng node.JS 14.x, trong khi server dùng node 8.x)
  - + Source code sử dụng nhiều cú pháp mới mà Node.JS chưa hỗ trợ (ví dụ sử dụng thư viện babel trong quá trình code)
- Máy chủ server thường rất mạnh mẽ (nhiều RAM/CPU) để phục vụ users

#### 2. Chạy Project local

##### Link projects:

<https://drive.google.com/file/d/12OJtv61QfTfmQQ2wB6utwvjLw4kKZnfz/view?usp=sharing>

Tài liệu: <https://github.com/babel/example-node-server>  
<https://babeljs.io/setup#installation>

## #20. Phân Biệt Development và Production

Tài liệu:

[https://expressjs.com/en/advanced/best-practice-performance.html#set-node\\_env-to-production](https://expressjs.com/en/advanced/best-practice-performance.html#set-node_env-to-production)

### 1.Applications Mode

Mặc định, khi chạy ứng dụng tại localhost, chúng ta quan niệm môi trường phát triển ứng dụng là 'development', gọi tắt là dev, vì đơn giản, nó dùng cho developer.

Khi chạy ứng dụng thực tế (ứng dụng chạy cho người dùng sử dụng), chúng ta quan niệm đây là môi trường production, gọi tắt là prod.

Hiểu đơn giản:

- Môi trường dev: dành cho developer phát triển ứng dụng (data sử dụng là data fake), chạy tại localhost.
- Môi trường prod: dành cho khách hàng trải nghiệm ứng dụng (data sử dụng là data thật, ứng dụng đã go-live), chạy với tên miền và hosting.

Việc phân chia mode, giúp ứng dụng có nhiều môi trường để phát triển, phục vụ các mục đích khác nhau.

Ví dụ: môi trường dev, kết nối tới database dev => thêm, sửa, xóa thoải mái.

Môi trường production, kết nối tới database prod => kiểm soát chặt chẽ việc thao tác

### 2.Node ENV

Tài liệu:

<https://www.dynatrace.com/news/blog/the-drastic-effects-of-omitting-node-env-in-your-express-js-applications/>

**process.env.NODE\_ENV**

**app.get('env') // returns 'development' if NODE\_ENV is not defined**

**Npm i cross-env**

**"start": "cross-env NODE\_ENV=production nodemon --exec babel-node src/server.js"**

**Đối với ứng dụng server, việc chỉ rõ môi trường phát triển ứng dụng, giúp tối ưu hóa hiệu năng.**

## #21. Nodemon, Babel & devDependencies

### 1. Nodemon

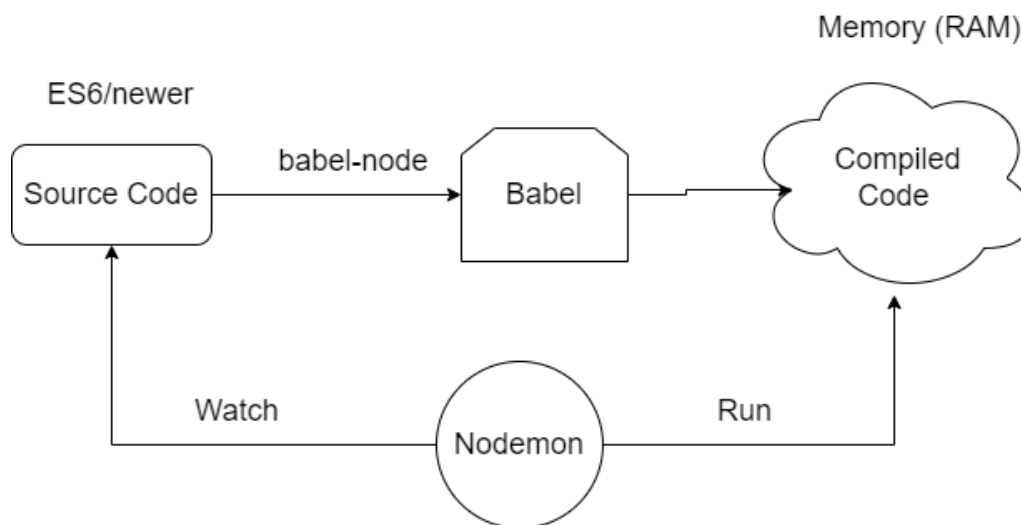
Là công cụ của developer, giúp auto chạy lại server mỗi khi có sự thay đổi của file.

### 2. Babel

**Babel là trình dịch code, giúp dịch code javascript về version mà 'node.js server' có thể chạy.**

Ví dụ: khi đã cấu hình babel -> sử dụng được cú pháp import, thay vì require

Quá trình này diễn ra như thế nào ?



### 3. devDependencies

**Là các thư viện sử dụng trong quá trình phát triển dự án -> phục vụ developer**

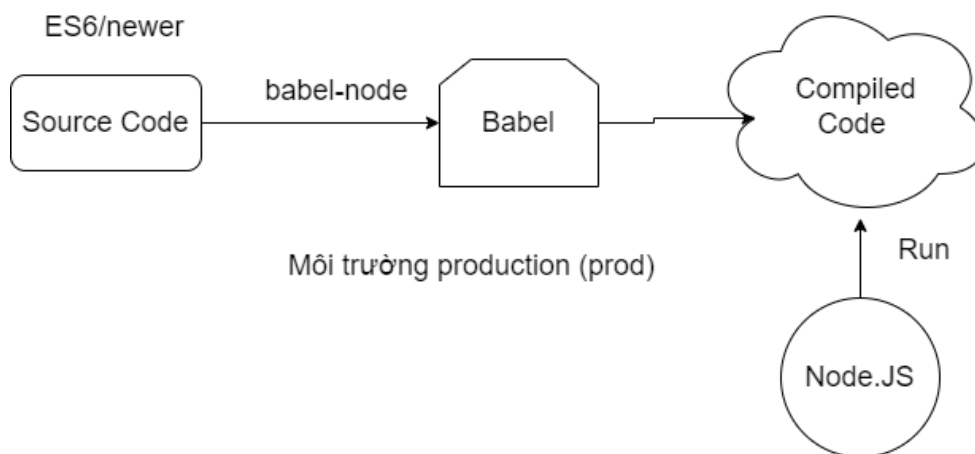
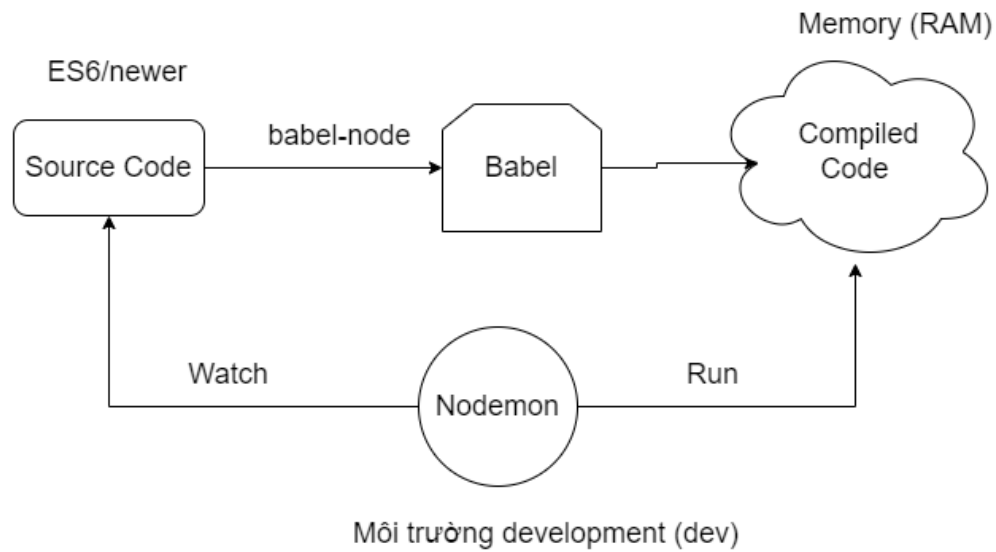
Khi build source code cho Production (chạy thực tế), các thư viện này sẽ không được cài đặt.

## #22. Babel Run Production

### 1. Tồn đọng của việc chạy code ở mode Development

Tối chưa được tối ưu:

- Nếu chạy code với babel -> không tối ưu performance



## 2. Build source code với Babel

Bước 1: dịch file javascript

**"build-src": "babel src -d build --copy-files"**

-d: tên thư mục output

--copy-files: copy non-javascript files (css, html...)

Bước 2: chạy source code => không cần nodemon => tối ưu hóa Memory

**"build": "node build/server.js"**

## #23. Run forever với PM2 (Auto Restart)

<https://expressjs.com/en/advanced/best-practice-performance.html#ensure-your-app-automatically-restarts>

<https://www.npmjs.com/package/pm2>

### 1. Auto restart

Ở chế độ dev, chúng ta dùng nodemon để auto restart server.

Tuy nhiên, ở mode production, chúng ta không sử dụng nodemon.

Ngoài ra, đôi khi, chúng ta muốn: server có lỗi => die => auto restart

```
app.get('/hoidanit', (req, res) => {  
  setTimeout(function () {  
    throw new Error("We crashed!!!!!!");  
  }, 10);  
})
```

### 2. PM2

Cài đặt : **npm install --save-exact pm2@5.2.2**

**npm install --save-exact pm2@5.2.2 -g**

Một vài package tương tự: forever, supervisord



## #24. Logging

### Tài liệu:

<https://expressjs.com/en/advanced/best-practice-performance.html#do-logging-correctly>

<https://reflectoring.io/node-logging-winston/>

### Sử dụng logging để ghi nhận:

- Requests (thông tin của request gửi lên server, thông thường, thông qua APIs)  
=> logs request information (header, body...)
- Changes: logs các lỗi xảy ra, các sự thay đổi xảy ra trong app (create/delete/update data)

### 1. Console.log/Console.error

console.log('some msg') will print msg to the standard output (stdout).  
console.error('some error') will print error to the standard error (stderr).

Các hạn chế:

- Không format (cấu trúc) được message hiển thị
- Không có logs level (debug/info/warning/error...)

Nếu, ứng dụng không có terminal để logs lỗi, hoặc trường hợp production, ứng dụng chạy lỗi hôm qua, ngày mai mới check, thì làm thế nào ?

### 2. Logging library

- Winston (phổ biến nhất)
- pino
- bunyan
- Morgan

### 3. Winston

<https://github.com/winstonjs/winston>

**npm install --save-exact winston@3.8.2**

//update logger.js

**Download file logger:**

<https://drive.google.com/file/d/199AJmp5SJUi5xWq7d75qOtvvAACXCm8i/view?usp=sharing>

//using

import logger from "../logger";

logger.log('info', 'Hello created log files!', { 'foo': 'bar' });

logger.info('Hello created log files!', { 'foo': 'bar' })

## #25. From Heroku to Render (Node.JS)

Heroku (founded 2007) là 1 platform giúp triển khai code trên cloud, hỗ trợ rất nhiều ngôn ngữ, trong đó có Node.JS

Tuy nhiên, từ tháng 12/2022, Heroku không có plan free, chỉ có plan trả phí (tối thiểu là 5\$/tháng)

Render (founded 2019), là 1 dịch vụ cloud cho hosting apps

Các lưu ý khi sử dụng Render:

- Giá cả: (có plan cho dùng free)

<https://render.com/pricing>

- Chi tiết của gói free: <https://render.com/docs/free>

### 1. Web service (backend node.js)

- Tự động hibernate sau 15 phút không hoạt động, có nghĩa là, nếu web server không có request nào gửi tới, thì nó 'ngủ'

Khi 1 request mới gửi tới => Render sẽ wake up (nhưng cần thời gian khoảng 30s) => tạo cảm giác loading (wait for...)

Mỗi tháng có 750 hours hoạt động miễn phí (tính cho tất cả services) => cái này reset every month

### 2. Database Postgres (free - cơ mà không xài)

Nhược điểm:

- Cho dùng thử trong vòng 90 days. sau khoảng thời gian này, không upgrade lên gói trả phí => say good bye  
database dùng thử sẽ bị xóa

- có thể tạo mới database khác rồi dùng tiếp => bất tiện

### 3. Tạo tài khoản render

Tạo được khoảng 3 projects free (khi không liên kết credit card)

Với render, chỉ dùng để host backend, còn database => host nơi khác

## **#26. Deploy with Render (Only Backend)**

chuẩn bị: push to source code to github

- kiểm tra project đã chạy tốt ở localhost hay chưa ?

- create render account

- create a project

- link render to github

- deploy (build)

## **#27. Create SQL Database (with supabase)**

Tại sao không dùng MySQL, mà lại dùng Postgres ?

Các bước cần làm:

- Tạo tài khoản

- Tạo project (lấy thông tin database)

- Test kết nối bằng phần mềm (ví dụ DBeaver)

- Test kết nối localhost (chạy app backend kết nối trực tiếp vào database này)

Link tham khảo: <https://www.youtube.com/watch?v=ZeuP2xEHTyI>

## **#28. Deploy Backend và SQL Database**

Làm tương tự như deploy mình app backend, chỉ khác chỗ cấu hình tham số môi trường để kết nối tới database

Link tham khảo: <https://www.youtube.com/watch?v=NkuIG9hH2LI>

## **#29. Nosql Database với Mongoddb Atlas**

Tham khảo #124 (tương tự khóa RestFul API)

## **#30. Deploy Backend và MongoDB**

Tham khảo #125 (tương tự khóa RestFul API)

### **#31. Nhận Xét Về Deploy Backend**

- Free ?
- performance ?

Khi sử dụng Hosting Backend Free, thường phải tìm hiểu cách host source code backend và Database

=> website thường chậm (tương tự như khi hosting với Frontend)

## Chapter 4: Docker và Nginx

### #32. Nginx Introduction

Ví dụ về website sử dụng Nginx: <https://www.airbnb.com/>

Nginx được sinh ra vào năm 2004, với mục đích giải quyết bài toán **C10k problem** (10.000 lượt kết nối cùng lúc tới 1 website)

Khoảng 40% website trên thế giới sử dụng Nginx để làm web server.

**Trong khóa học này, chúng ta sử dụng phiên bản mã nguồn mở của nginx**

Các tác dụng nổi bật của Nginx có thể kể tới như:

- Web server (chạy static file)
- Load balancing (giúp cân bằng tải)
- Reverse proxy (giúp che dấu thông tin service sử dụng)
- Caching data (giúp truy cập website nhanh hơn)
- ....

Tất cả các tính năng nổi bật của Nginx sẽ được đề cập cụ thể tại các video tiếp theo.

## #33. Nginx với Docker

### 1. Running image

[https://hub.docker.com/\\_/nginx](https://hub.docker.com/_/nginx)

- **pull image:** (kéo images từ internet về local)

`docker pull nginx` //pull latest version

`docker pull nginx:1.23`

- **kiểm tra image:**

`docker images`

hoặc check tab images của docker desktop

- **run image:**

<https://docs.docker.com/engine/reference/commandline/run/>

//tag 1.23

`docker run -it -d -p 8000:80 --name nginx-container nginx:1.23`

//tag latest

`docker run -it -d -p 8000:80 --name nginx-container nginx`

//bonus:

`docker run -it --rm -d -p 8000:80 --name nginx-container nginx`

- **kiểm tra running images:**

`docker ps` //show running containers

`docker ps -a` //show all

ps means 'Process Status'

### 2. Hello world

Go to: `http://localhost:8000/`

`curl localhost:8000`

### 3. Verifying your installation



VSCode extension:

<https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker>

choose container

Right click => attach shell

- check version

nginx -v

### **#34. Nginx Basic Command**

Tài liệu: [https://nginx.org/en/docs/beginners\\_guide.html](https://nginx.org/en/docs/beginners_guide.html)

VSCode extension:

<https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker>

choose container

Right click => attach shell

- Thao tác trực tiếp với nginx, thông qua docker

- stop nginx:

nginx -s stop

service nginx stop

- start nginx => run docker :v

nginx -h //help

nginx -v -V

nginx -t -T //show config + test

### **#35. Nginx Default Folder**

default folder: /usr/share/nginx/html

Chọn container => Files => /usr/share/nginx/html

modify index.html

## **#36. Nginx với Docker Compose**

### **Download source code:**

[https://drive.google.com/drive/folders/1c\\_HQx0RZJ6thmNrL6pq1N4TJo63Fve6y?usp=sharing](https://drive.google.com/drive/folders/1c_HQx0RZJ6thmNrL6pq1N4TJo63Fve6y?usp=sharing)

- prepare docker compose folder

- only html files (volume)

- run inside compose folder

- stop current nginx

docker compose -p hoidanit up -d

go to localhost:8000 => forbidden . oh yeah

### **#37. Nginx Customize HTML**

**Download source code:**

[https://drive.google.com/drive/folders/1qX68qYyKbqMK5uOUSG5CyoXnnwysJSgO?usp=share\\_link](https://drive.google.com/drive/folders/1qX68qYyKbqMK5uOUSG5CyoXnnwysJSgO?usp=share_link)

- using volume  
create index.html

### **#38. Run React App With Nginx (Basic)**

**Download source code:**

[https://drive.google.com/drive/folders/1ouegbW7Vhou88qe0j\\_3LolrwmG9vnHpN?usp=share\\_link](https://drive.google.com/drive/folders/1ouegbW7Vhou88qe0j_3LolrwmG9vnHpN?usp=share_link)

- ./build/:/usr/share/nginx/html/

build xong bị lỗi khi refresh

cần update file config

## **Chapter 5: Nginx Web Server**

### **#39. Nginx Config Files**

nginx -T

/etc/nginx/nginx.conf

/etc/nginx/conf.d/default.conf

===

clone file setup default => sau đấy sử dụng docker compose (with empty)

chỉ cần ghi đè /etc/nginx/nginx.conf , vì file default.conf được import vào.

## #40. Syntax Config Files

Tài liệu: <http://nginx.org/en/docs/dirindex.html>

<https://docs.nginx.com/nginx/admin-guide/basic-functionality/managing-configuration-files/>

- Để test file config tự viết, sử dụng: `nginx -t`
- Để biết file config lưu ở đâu + test: `nginx -T`

### 1. Directives (chỉ dẫn)

```
//angular  
<input [(ngModel)]="currentItem.name" id="example-ngModel">
```

Directives là các chỉ dẫn giúp 'công cụ' nó hiểu chúng ta viết gì. nginx không ngoại lệ, cần viết theo cú pháp của nó.

ví dụ:

```
user      nobody;  
error_log logs/error.log notice;  
worker_processes 1;
```

**lưu ý: với directives, kết thúc bằng dấu chấm phẩy (;)**

### 2. Block/Context

- Để nhóm các directives liên quan tới nhau => sử dụng curly braces ({ })

A few top-level directives, referred to as contexts, group together the directives that apply to different traffic types:

```
events – General connection processing  
http – HTTP traffic  
mail – Mail traffic  
stream – TCP and UDP traffic
```



=====

```
http {  
    # Configuration specific to HTTP and affecting all virtual servers  
    server {  
        # configuration of HTTP virtual server 1  
        location /one {  
            root /usr/share/nginx/html;  
        }  
        location /two {  
            # configuration for processing URIs starting with '/two'  
        }  
    }  
}
```

## **#41. Setting Up Virtual Servers**

<https://docs.nginx.com/nginx/admin-guide/web-server/web-server/>

Lưu ý: tạo file backup của config gốc lấy cái tham chiếu

```
http
server
location
    root
    return
```

## **#42. MIME Types**

[http://nginx.org/en/docs/http/nginx\\_http\\_core\\_module.html#types](http://nginx.org/en/docs/http/nginx_http_core_module.html#types)

[https://developer.mozilla.org/en-US/docs/Web/HTTP/Basic\\_of\\_HTTP/MIME\\_types/Common\\_types](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basic_of_HTTP/MIME_types/Common_types)  
[https://developer.mozilla.org/en-US/docs/Web/HTTP/Basic\\_of\\_HTTP/MIME\\_types/Common\\_types](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basic_of_HTTP/MIME_types/Common_types)

## **#43. Handling Errors**

<https://docs.nginx.com/nginx/admin-guide/web-server/web-server/#handling-errors>

phân biệt location = / và location /

```
location / {  
    root /usr/share/nginx/html;  
    index index.html index.htm;  
}
```

#### **#44. Run React App With Nginx (Advance)**

Link download source code thực hành:

[https://drive.google.com/drive/folders/1-Bwi6rzLDpqvOzFgZx5fXLnL376oslx5?usp=share\\_link](https://drive.google.com/drive/folders/1-Bwi6rzLDpqvOzFgZx5fXLnL376oslx5?usp=share_link)

-trong thực tế có cần dùng customize 404 not found và 50x ko ?

- fix lỗi react f5

#### **#45. Giới thiệu Serving Static Content**

Tài liệu: <https://docs.nginx.com/nginx/admin-guide/web-server/serving-static-content/>

#### **#46. Giới thiệu Compression và Decompression**

Tài liệu: <https://docs.nginx.com/nginx/admin-guide/web-server/compression/>

#### **#47. Nhận xét về web server (only Frontend)**

Tương tự như việc deploy Frontend ở các chương trước, website không có backend sẽ chỉ hiển thị được thông tin tĩnh (static)

## **Chapter 6: Nginx Reverse Proxy & Load Balancing**

### **#48. Phân biệt Proxy và Firewall**

Tài liệu:

<https://www.l7defense.com/cyber-security/firewall-vs-proxy-server/>

Tóm gọn sự khác biệt giữa Proxy và Firewall:

Proxy: che đi danh tính thực sự của bạn (IP address)

Firewall: kiểm soát (định danh bạn là ai và có quyền truy cập gì) khi sử dụng website và các hình thức truy cập tương ứng (http, https, ftp, ftps...)

### **#49. Phân biệt Reverse Proxy và Forward Proxy**

**Forward Proxy:** forward request (dấu đi định danh của người dùng), hay được dùng khi user muốn truy cập 1 nội dung trên internet

**Reverse Proxy:** được dùng cho webserver, với tác dụng load balancing, caching data và hide IP của server đích mà người dùng muốn truy cập

## **#50. Ví dụ App React Với Proxy**

Source code video:

[https://drive.google.com/drive/folders/18M2swx-jhbvi8GU27WR88ja-MoKT9dZN?usp=share\\_link](https://drive.google.com/drive/folders/18M2swx-jhbvi8GU27WR88ja-MoKT9dZN?usp=share_link)

## **#51. Build React App với Nginx và Backend**

=> có api

App backend này nhả ra apis để app frontend dùng

## **#52. Nginx Proxy Pass**

<https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/>

<https://stackoverflow.com/questions/27810076/how-do-i-access-a-server-on-localhost-with-nginx-docker-container>

```
location / {  
    proxy_pass http://host.docker.internal:3000;  
}
```

### **#53. Load Balancing: upstream**

<https://www.nginx.com/blog/inside-nginx-how-we-designed-for-performance-scale/>

<https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/>

apt-get update && apt-get install procps

ps -ef --forest | grep nginx

### **#54. Setup Docker Compose Backend và Database Service**

Link source code:

[https://drive.google.com/file/d/15V-std7XqR9V2PulOMrmVmlfbgS8DmY/view?usp=share\\_link](https://drive.google.com/file/d/15V-std7XqR9V2PulOMrmVmlfbgS8DmY/view?usp=share_link)

### **#55. Setup Docker Compose React và Nginx**

Link source code:

[https://drive.google.com/file/d/1\\_hJUzz2X7KKlxAYYsJroatD0a1Ujrgtn/view?usp=share\\_link](https://drive.google.com/file/d/1_hJUzz2X7KKlxAYYsJroatD0a1Ujrgtn/view?usp=share_link)

Build react + nginx

docker system prune -af

### **#56. Run Full App với Docker**

Sử dụng file default của nginx => ghi đè lại file này

Link source code:

[https://drive.google.com/file/d/183DMURdEjbj0s5vGFofy4G89Z7CedAuC/view?usp=share\\_link](https://drive.google.com/file/d/183DMURdEjbj0s5vGFofy4G89Z7CedAuC/view?usp=share_link)

### **#57. Bỏ trợ - Docker Compose Dạng Basic (Mapping port)**

Sử dụng mapping port, ko expose port nữa :v

Link source code:

[https://drive.google.com/file/d/1rzAdX47ZvJopkO9c6OR29apUb1qioQIT/view?usp=share\\_link](https://drive.google.com/file/d/1rzAdX47ZvJopkO9c6OR29apUb1qioQIT/view?usp=share_link)



## Chapter 7: Run Web App với Hosting/Domain

### #58. Hosting Server Website

#### 1. Hosting website

Tài liệu:

<https://wiki.matbao.net/hosting-la-gi-tong-hop-tat-ca-thong-tin-can-biet-khi-mua-hosting>

Hosting là cách chạy source code trên máy chủ server.

Máy chủ server là 1 máy tính hoạt động 24/7, và có thể truy cập từ ngoài internet.

Có 2 loại hosting:

- miễn phí (render.com/heroku.com...)
- trả phí.

Mua hosting, thực chất là mua 1 địa chỉ IP của máy chủ server (cần địa chỉ IP thì mọi người mới kết nối internet tới được).

Các tiêu chí cần quan tâm khi mua hosting:

- Hệ điều hành
- Nhà cung cấp

#### 2. Hệ điều hành cho hosting (server)

Hệ điều hành dành cho máy chủ server khác với hệ điều hành dành cho người dùng phổ thông.

Với máy chủ server, hệ điều hành sẽ được tối ưu hóa hơn (giảm UI, tăng OS). có nghĩa là giao diện có thể xấu, nhưng hiệu năng và tính bảo mật cao

Có 2 hệ điều hành chính được sử dụng làm server: Linux và Windows (MacOS không được sử dụng nhiều)

Với linux server, các OS được sử dụng phổ biến:

- CentOS
- Ubuntu
- Debian

Với windows, có windows server

**Việc lựa chọn hệ điều hành nào, là phụ thuộc vào người sử dụng (kinh nghiệm sử dụng)**

### 3. Nhà cung cấp hosting

Có 2 loại, trong nước và ngoài nước (quốc tế)

- trong nước:

- + hình thức thanh toán nội địa (qua thẻ ngân hàng)
- + support tiếng việt
- + ko lệch múi giờ
- + giá thành cao hơn

- ngoài nước:

- + hình thức thanh toán qua thẻ Visa/master
- + support tiếng anh
- + lệch múi giờ (đôi khi sẽ có sự delay khi support)
- + giá thành rẻ hơn

Với người mới bắt đầu, ít kinh nghiệm => chọn hosting trong nước (được tư vấn và support phù hợp với nhu cầu)

### 4. Mua hosting

**(phần mua hosting mình sẽ hướng dẫn ở video tiếp theo)**

Tham khảo:

<https://quantrimang.com/cong-nghe/top-nha-cung-cap-hosting-toc-do-cao-va-bao-mat-tot-nhat-177077>

- Các bước thực hiện:

1. Chọn nhà cung cấp dịch vụ
2. Xác định nhu cầu sử dụng (ít hay nhiều, dùng để test học tập hay kinh doanh, dành cho cá nhân hay doanh nghiệp...)
3. Chọn plan phù hợp
4. Thanh toán và sử dụng.

**Output: Có được tài khoản đăng nhập và địa chỉ IP của server**

## #59. Hướng Dẫn Mua Hosting

### 1. Chọn cấu hình Hosting phù hợp

Có 3 loại hosting chủ yếu:

- **Hosting với server đã cấu hình sẵn (cloud hosting)**, chỉ cần upload code lên rồi chạy. Cách làm này thường được dùng nhất với Wordpress. Chúng ta không dùng loại này, vì không thể tự cài đặt công cụ cần thiết.  
Mình ví dụ: cài mongodb, cài redis, cài postgres... nó không cho, nó bắt dùng mysql chẳng hạn. Muốn cài thêm docker lại càng xa vời  
=> loại host này phù hợp với website được xây dựng với wordpress
- **Hosting với server (cloud server)**: loại này ngon, có mỗi tội là đắt, phù hợp với doanh nghiệp. Chúng ta có thể cài đặt tùy thích trên server
- **Hosting với VPS (virtual private server)** : loại này rẻ, phù hợp với nhu cầu vừa và nhỏ

Trong khóa này, **để tiết kiệm chi phí**, mình gợi ý các bạn dùng của nhà cung cấp vietnx  
Tham khảo chi phí tại đây: <https://vietnix.vn/cloud-vps-gia-re/>

**Với nhà hosting này, bạn có thể dùng plan 1 tháng, giá từ 89k/tháng** test cấu hình website các kiểu, xem nó có ok không, nếu ok thì đăng ký dùng lâu dài, còn nếu không ok đổi gói khác sao cho phù hợp với nhu cầu của bạn

Cá nhân mình, hiện tại, như website hoidanit.com.vn, mình đang sử dụng của mắt bão, giá 259k/tháng : <https://www.matbao.net/server/cloud-server.html>

=> **các bạn không nhất thiết phải mua như mình** (vì cá nhân mình dùng server với nhiều mục đích khác nhau).

=> **không có khái niệm nhà cung cấp nào là tốt nhất** (giống hệt như kiểu chọn nhà cung cấp dịch vụ mạng internet ấy :v )

### Yêu cầu của phần mua hosting này:

Cần có 1 tài khoản để login vào server thông qua ssh, **cấu hình tối thiểu của server:**

CPU 1 core

RAM 1GB

Disk 10GB

**Nếu nhiều hơn thì càng tốt các bạn nhé :v**

## 2. Một vài lưu ý khi mua Hosting giá rẻ tại Vietnx

Sau khi mua hosting, thông tin server sẽ được gửi qua email (gồm địa chỉ ip, username/password, link đăng nhập trang quản trị server...)

Mặc định, chúng ta cần có “hợp đồng” khi mua hosting, và cần nộp trong vòng 90 ngày (các bạn check email, có cái email ghi vậy). Tuy nhiên, chúng ta “cứ kệ”, vì đang dùng server này để test.

Nếu test thấy server ok, chạy không bị quá tải (RAM/CPU), đảm bảo được hiệu năng cho website, các bạn nên chủ động liên hệ tổng đài để hỏi rõ về thủ tục nhé, đảm bảo quyền lợi của khách hàng

## #60. Công cụ kết nối vào Hosting

### 1. Thao tác qua giao diện

Với windows server, có thể sử dụng giao diện server để thao tác thông qua chức năng 'remote desktop' của windows.

Việc thao tác này giống như việc dùng Teamviewer/Anydesk... để kết nối vào máy chủ server thông qua địa chỉ IP và user/password.

Về chức năng remote desktop của windows:

tham khảo video: <https://www.youtube.com/watch?v=LmnMRCixwLU>

xem từ 5:00

video trên, phần đầu là setup 1 máy ảo windows, sau đấy kết nối vào máy tính ảo đấy qua remote desktop.

### 2. Thao tác qua terminal (hay dùng)

Với máy tính Linux/MacOS => dùng sẵn terminal đã có.

Với windows:

- Phần mềm Windows Terminal:

<https://www.microsoft.com/store/apps/9n0dx20hk701>

windows Store => Windows Terminal

Sau khi cài đặt, mở Windows Terminal. Login vào server bằng câu lệnh:

```
ssh your-user@VPS-IP-address
```

ví dụ: `ssh root@192.168.1.69`

Ngoài Windows Terminal, có thể sử dụng gitbash, hoặc một vài phần mềm khác, ví dụ: Xshell...

**Phần mềm giả lập môi trường windows: (copy file từ windows vào linux)**

WinSCP: <https://winscp.net/eng/download.php>

FileZilla: <https://filezilla-project.org/>

## **#61. Sử Dụng FileZilla (MacOS)**

Link download:

[https://filezilla-project.org/download.php?show\\_all=1](https://filezilla-project.org/download.php?show_all=1)

## #61.1 Cấu Hình Hosting (Basic)

Lưu ý: với hosting giá rẻ (VPS) của vietnx, server được tạo không có Apache và đã cài đặt sẵn firewall, thành ra các bước bên dưới sẽ có một chút khác biệt (tuy nhiên không đáng kể)

### 1. Check đăng nhập với SSH và đổi mật khẩu/tạo user

**Lưu ý: để an toàn tối đa, nên tìm hiểu về đăng nhập thông qua SSH key (private/public key)**

Cần đổi mật khẩu mặc định, thành mật của bạn (lưu ý nên đặt mật khẩu có chữ hoa, chữ thường và có ký tự đặc biệt, không nên đặt mật khẩu quá yếu)

Thay đổi mật khẩu cho user root:

**sudo passwd root**

Tạo user mới, và gán quyền root cho user (sudo)

**adduser USERNAME**

Assign: **usermod -aG sudo USERNAME**

Check: **groups USERNAME**

### 2. Remove Apache (Nếu server của bạn đã cài sẵn Apache)

Mặc định (thông thường) các server Ubuntu đã cài đặt sẵn Apache  
=> remove để dùng cho Nginx.

Nếu server của bạn không cài đặt apache (gõ địa chỉ ip vào browser enter không ra gì, hoặc gõ **systemctl status apache2 không ra kết quả**) không cần sử dụng mục 2 này, chuyển qua mục 3.

Kiểm tra apache:

**systemctl status apache2**

Stop apache:

**systemctl stop apache2**

Delete apache server:

**systemctl disable apache2**

**apt remove apache2**

//delete related dependencies

**apt autoremove**

//Cleaning and updating server

**apt clean all && sudo apt update && sudo apt dist-upgrade**

**rm -rf /var/www/html**

### 3.Installing and configure Firewall

**Tài liệu:**

<https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-ubuntu-18-04>

<https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands>

- set up ufw (firewall)

**apt install ufw**

- check status

**ufw status**

- check apps:

**ufw app list**

- allow ssh:

**ufw allow ssh**

- tạo config port cho nginx: (tương tự sao này muốn nhiều mở nhiều port khác)

**ufw allow 80**

**ufw allow 443**

**OR**

**ufw allow http**

**ufw allow https**

- enable firewall:

**ufw enable**

- disable/remove

**ufw disable**

**apt remove ufw**



## #61.2. Setup Docker với Hosting

Tài liệu:

Version 22.04

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-22-04>

Version: 18.04

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04>

### 1.setup docker và docker compose

First, update your existing list of packages:

**sudo apt update**

Next, install a few prerequisite packages which let apt use packages over HTTPS:

**sudo apt install apt-transport-https ca-certificates curl software-properties-common**

Then add the GPG key for the official Docker repository to your system:

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**

Add the Docker repository to APT sources:

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"**

Next, update the package database with the Docker packages from the newly added repo:

**sudo apt update**

Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

**apt-cache policy docker-ce**

docker-ce:

Installed: (none)

Candidate: 18.03.1~ce~3-0~ubuntu

Version table:

18.03.1~ce~3-0~ubuntu 500

500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages

Finally, install Docker (and docker compose):

**sudo apt install docker-ce docker-compose-plugin**

**Verify cài đặt docker:**

sudo docker version

sudo docker compose version

## 2. Bỏ keyword sudo khi run docker

If you want to avoid typing sudo whenever you run the docker command, add your username to the docker group:

**sudo usermod -aG docker \${USER}**

To apply the new group membership, log out of the server and back in, or type the following:

**su - \${USER}**

If you need to add a user to the docker group that you're not logged in as, declare that username explicitly using:

**sudo usermod -aG docker username**

## 3. Test docker với nginx

**docker run --name hoidanit -p 80:80 -d nginx**

**docker stop hoidanit**

**docker system prune -af**

## **#62. Run app with Hosting**

copy files

run docker compose

done :v

check available storage (disk space): df -h

<https://itsfoss.com/check-free-disk-space-linux/>

check cpu/ram: htop

<https://geekflare.com/process-cpu-memory-monitoring/>

## **#63. Check IP Client (Docker Logs)**

Tài liệu: <https://docs.docker.com/engine/reference/commandline/logs/>

## #64. Tên miền Website (Domain)

### 1. Domain là gì

Mỗi 1 website (server), có 1 địa chỉ IP duy nhất (unique), giúp người dùng internet có thể truy cập mọi lúc mọi nơi.

Tuy nhiên, địa chỉ IP (ví dụ 192.168.6.9) là 1 dãy số khó nhớ => tên miền được sinh ra.  
Ví dụ: google.com dễ nhớ hơn địa chỉ 192.x.y.z

Quá trình sử dụng website của client:

- Mở trình duyệt web (browser) và gõ tên miền google.com => nhấn enter
- Nhà mạng sẽ mapping tên miền vào địa chỉ IP tương ứng (dựa vào DNS: Domain Name System)

sau quá trình mapping này => client sẽ truy cập gián tiếp vào IP của server.

### 2. Mua tên miền

- Tương tự việc dùng mạng internet, cần mua tên miền (các nhà mạng sẽ làm bước mapping tên miền vào địa chỉ IP server)

Tương tự việc mua hosting, với domain cũng tương tự, phụ thuộc vào nhu cầu sử dụng để lựa chọn nhà cung cấp:

- trong nước
- ngoài nước

Với website phục vụ chủ yếu cho thị trường việt nam => nên chọn nhà cung cấp trong nước (support 24/7, dễ dàng re-new)

## #65. Mua Tên Miền

### 1. Tạo tài khoản

Trong khóa học này, mình hướng dẫn mua tên miền của nhà cung cấp Mắt Bão

<https://www.matbao.net/>

Khi mua tên miền, có thể mua trước rồi tạo tài khoản sau.

Với lần đầu tiên mua tên miền thành công, thông tin tài khoản sẽ được gửi qua email đăng ký (username đăng nhập/password)

### 2. Mua tên miền

**Lưu ý về tên miền quốc tế và tên miền việt nam:**

Với tên miền quốc tế, giá cả sẽ thường đắt hơn (ví dụ .com) so với tên miền trong nước, và có nhiều loại rẻ hơn.

**Tuy nhiên, mình khuyến khích mua tên miền trong nước** (ví dụ .vn, .net, .com.vn...), vì những tên miền này được bảo hộ và có quyền sở hữu cao hơn so với tên miền quốc tế

**Hiểu đơn giản là, với tên miền trong nước, khả năng bị mất tên miền là khá thấp khi bạn đã mua thành công tên miền đó**

**Trong quá trình mua tên miền, cần điền thông tin thật, vì đây là cơ sở để chứng minh quyền sở hữu hợp pháp của bạn :v**

## **#66. Điều Cần Làm Sau Khi Mua Domain**

### **1. Tạo mật khẩu đăng nhập dashboard**

Sau khi mua tên miền thành công, bạn sẽ nhận được email thông báo đổi mật khẩu. Vì vậy, nên đăng nhập vào trang quản lý của hosting và tiến hành đổi mật khẩu cho tài khoản của bạn

### **2. Vấn đề bảo mật tài khoản (2 auth factor)**

Nên tìm hiểu phần settings cho tài khoản, và nếu được, nên bật xác thực 2 lớp để tăng cường tính bảo mật cho tài khoản của bạn

### **3. Update thông tin**

Bạn cần cập nhật thông tin tài khoản (sử dụng CCCD) để xác thực danh tính.

Ngoài ra, khi mua tên miền thành công, có một vài thủ tục (hợp đồng) cần hoàn thiện, truy cập trang quản lý để biết chi tiết.

**Nếu bạn không biết nên làm như thế nào, thì hãy gọi bộ phận hỗ trợ của bên dịch vụ nhé (thường là support 24/7)**

## **#67. Mapping Domain tới Hosting**

<https://wiki.matbao.net/kb/huong-dan-quan-ly-ten-mien-tai-mat-bao/>

<https://wiki.matbao.net/kb/huong-dan-cau-hinh-ten-mien-ve-hosting/>

- DNS:

<https://wiki.matbao.net/kb/quan-ly-ban-ghi-dns/>

<https://youtu.be/grnuaFMl1q>

<https://www.site24x7.com/learn/dns-record-types.html>

type A. mapping address => ip (v4)

AAAA                      ip (v6)

CNAME (canonical name) point domain to another domain



## **#68. Mapping Domain tới Github Page**

- lưu ý cần build lại source code với vite => update base path

<https://vitejs.dev/guide/static-deploy.html#github-pages>

**github page => update file 404 not found**

**var pathSegmentsToKeep = 0;**

## Chapter 8 : Securing The Apps

### #69. Setup Run Docker (without sudo)

Tài liệu:

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-22-04#step-2-executing-the-docker-command-without-sudo-optional>

#### 1. Môi trường

- Ubuntu
- Đã cài đặt thành công Docker (và Docker compose)

Mặc định, câu lệnh docker chỉ có thể chạy dưới quyền user root, thông qua keyword sudo

#### 2. Docker without sudo

**Áp dụng cho user đang login vào host :**

- Add user vào 'docker group'

**sudo usermod -aG docker \${USER}**

- Apply user vào group trên:

**su - \${USER}**

- check xem user đã được add vào docker chưa:

**groups**

- restart system:

**sudo reboot**

sau câu lệnh trên, host sẽ bị restart (connection sẽ disconnect), cần chờ 1 chút để có thể login lại

- bonus :

stop container running & images:

**docker stop \$(docker ps -aq) && docker rm \$(docker ps -aq)**

xóa all:

**docker system prune -af**

kiểm tra kết quả với:

**docker ps**

**docker images**

- xóa folder ubuntu: **sudo rm -rf folder\_name**

<https://askubuntu.com/questions/217893/how-to-delete-a-non-empty-directory-in-terminal>

- kill port: **sudo kill -9 \$(sudo lsof -t -i:PORT)**

ví dụ: **sudo kill -9 \$(sudo lsof -t -i:80)**

-----

Xóa người dùng ra khỏi group: (làm ngược lại video này)

=> cần xóa ra khỏi group docker

**deluser <username> <groupname>**

Sau đấy: **sudo reboot** (cho chắc ăn :v)

Ví dụ: **sudo deluser hoidanit docker**

## #70. SSL Certificate

### 1. SSL là gì

<https://www.hostinger.com/tutorials/what-is-ssl>

Hiểu 1 cách đơn giản, SSL giúp việc truy cập và sử dụng website an toàn hơn cho người dùng, "hạn chế" được tấn công của các hacker.

SSL giúp thông tin giữa req và res được encrypt/decrypt

Các giao dịch về tiền tệ, thông tin người dùng, nếu không được encrypt/decrypt sẽ rất nguy hiểm

Website có SSL sẽ giúp tăng thứ hạng tìm kiếm trên Google (SEO ranking)

- cách check SSL trên browser

ví dụ website ko có SSL: [http://nginx.org/en/docs/http/configuring\\_https\\_servers.html](http://nginx.org/en/docs/http/configuring_https_servers.html)

có ssl: <https://www.nginx.com/>

### 2. SSL Certificate

- ssl cần gia hạn ?

SSL sau 1 thời gian sử dụng nhất định, cần gia hạn (re-new) để đảm bảo tính an toàn

- trả phí hay miễn phí ?

Có thể tự issue 1 certificate miễn phí (với openssl)

=> lên production dùng certificate do các tổ chức cung cấp (CA - Certificate Authority )

Ví dụ: <https://www.digicert.com/kb/ssl-support/openssl-quick-reference-guide.htm>

(với windows, dùng gitbash để chạy)

Trong thực tế, khi chạy production, nên dùng certificate do CA cung cấp => tăng tính bảo mật và độ an toàn

Trường Issued by => nơi mua certificate

<https://www.amazon.com/>

<https://github.com/>

=> DigiCert Inc

<https://www.digitalocean.com/>

=> Cloudflare Inc

<https://www.nginx.com/>

=> Let's Encrypt

**Trả phí:**

- Support (24/7)
- Phù hợp với mọi level

**Miễn phí:**

- Support (community)
- cá nhân/ small business

## #71. Let's Encrypt và Certbot

### 1. Let's Encrypt

Tài liệu: <https://letsencrypt.org/>

- Let's Encrypt là CA - Certificate Authority

1 tổ chức phi lợi nhuận (non profit) giúp chứng nhận và cung cấp chứng chỉ TLS (SSL) cho website hoàn toàn miễn phí.

- Đã cung cấp SSL cho hơn 300M websites

### Quy trình cấp SSL với Let's Encrypt:

1. User request Certificate (cung cấp email, tên miền) tới Let's encrypt

2. Nhận về certificate

3. Run website (port 443) với certificate

Thông thường certificate sẽ hết hạn sau 90 ngày.

=> cần renew certificate

### 2. Certbot

Tài liệu: <https://certbot.eff.org/>

<https://eff-certbot.readthedocs.io/en/stable/>

- Certbot là công cụ giúp 'tự động' cấu hình (và re-new) TLS/SSL cho Apache và Nginx

- Được sử dụng tại web server (hosting), chạy tại PC (personal computer) không có ý nghĩa.

- Có thể sử dụng certbot như 1 phần mềm (thông qua Snap) hoặc sử dụng công cụ khác, ví dụ như Docker.

## #72. Run Nginx với SSL (Basic)

Link download source code:

<https://drive.google.com/file/d/1IZIPxIndQ8pzjVKfocFfhUdyK18droa6/view?usp=sharing>

### 1. Một vài lưu ý

Ý tưởng của project được lấy từ github: <https://github.com/wmnnd/nginx-certbot>

Tuy nhiên, project trên đã tạo từ lâu, và do version của phần mềm thay đổi theo thời gian, nên khi sử dụng images của certbot và nginx từ docker hub, cần check đúng version (không nên lấy version là latest)

Certbot là công cụ giúp lấy SSL certificate từ Let's encrypt, tuy nhiên, với Let's encrypt, nó giới hạn số lần gọi API, chi tiết xem tại đây:

<https://letsencrypt.org/docs/rate-limits/>

<https://letsencrypt.org/docs/staging-environment/>

Để tránh tình trạng bị limit số lần gọi request, trong môi trường test, cần set biến staging = 1 (như trong video hướng dẫn)

### 2. Các bước thực hiện

- **Đảm bảo rằng máy tính của bạn đã chạy docker - docker is running** (nếu không sẽ bị lỗi docker daemon is not running)
- Nếu được, thì nên xóa hết các images không dùng : `docker system prune -af`
- Chạy file `init-letsencrypt.sh`  
Để chạy file này (với windows), **sử dụng gitbash và gõ: `sh init-letsencrypt.sh`**

Với macos (or linux): cần set quyền rồi mới chạy (lưu ý chạy file nơi lưu file)

**`chmod +x init-letsencrypt.sh`**

**`bash init-letsencrypt.sh`**

### #73. Run Nginx full App với SSL (Staging)

Link source code:

[https://drive.google.com/file/d/1BxCxcAI3eP\\_lvP05vOcNGvgF36Whuwqe/view?usp=share\\_link](https://drive.google.com/file/d/1BxCxcAI3eP_lvP05vOcNGvgF36Whuwqe/view?usp=share_link)

Cách convert file sh từ windows sang ubuntu:

<https://stackoverflow.com/questions/27176781/bash-file-returns-unexpected-token-dollar>

Bước 1: Cài đặt dos2unix

```
sudo apt update
```

```
sudo apt install -y dos2unix
```

Bước 2: convert file sh

```
dos2unix file_name.sh
```

### **Lưu ý về images của certbot ứng với server DNS:**

Trong khóa học này, mình mua tên miền ở Mắt Bão (với server DNS là ns1), vì vậy image **của certbot mình chọn là dns-nsone**

Nếu bạn mua Domain ở nơi khác, thì chọn plugin tương ứng nhé

Full list support gồm:

dns-dnsmadeeasy

dns-dnssimple

dns-ovh

dns-cloudflare

dns-cloudxns

dns-digitalocean

dns-google

dns-luadns

dns-rfc2136

dns-route53

dns-gehirn



dns-linode

dns-sakuracloud

Tham khảo thêm tại đây:

<https://hub.docker.com/r/certbot/certbot>

<https://eff-certbot.readthedocs.io/en/stable/using.html#dns-plugins>

#### **#74. Check Renew Certificate**

Trong khóa học, mình đã setup certbot auto renew rồi. Thông thường, SSL do Lets encrypt sẽ có thời hạn 3 tháng. Sau 3 tháng sẽ có kết quả :v

#### **#75. Run Nginx full App với SSL (Production)**

Set staging = 0

Cách convert file sh từ windows sang ubuntu:

<https://stackoverflow.com/questions/27176781/bash-file-returns-unexpected-token-do-r>

Bước 1: Cài đặt dos2unix

`sudo apt update`

`sudo apt install -y dos2unix`

Bước 2: convert file sh

`dos2unix file_name.sh`

## #76. Các bước để tự chạy ứng dụng của bạn

Để tự build và deploy ứng dụng của bạn, các bước cần làm như sau (đã có domain và hosting)

### 1. Bước 1: Build Docker File

Bạn cần build từng thành phần của ứng dụng thông qua docker. Ví dụ như build ứng dụng frontend, rồi ứng dụng backend và các service cần thiết (như database chẳng hạn)

Quá trình này nó không khó đến vậy đâu. Cần gì cứ google : tên frameword/công cụ và thêm keyword docker file là nó ra đây

Sau khi đã build được ứng dụng gồm FE/BE và database, bây giờ ghép thêm nginx vào. Nginx sẽ chịu trách nhiệm chạy FE, đồng thời kết nối với BE.

Kết thúc bước này là chạy thành công tại localhost full ứng dụng (với http)

Test ngon lành thì sang bước tiếp theo, tích hợp certbot và SSL fake.

### Bước 2: Chạy với SSL fake

Ở đây, cần test tại local xem ứng dụng đã chạy được với SSL chưa, nếu chạy ok thì chuyển qua bước cuối

### Bước 3: Chạy với SSL thật

Copy code (hoặc bạn nào thích thì cài thêm git, clone code về server hosting).

Sau đấy, chạy câu lệnh init để tạo SSL thật.

Restart lại Nginx để nhận SSL mới => Done

## **#77. Nhận xét về Build & Deploy Website từ A tới Z**

- Liệu học xong, **các công ty ko cần tuyển người** ? (wordpress...)
- Web lớn thì nó khác gì ?
- Jmeter test performance
- Ưu nhược điểm của cách làm hiện tại  
Nhược điểm: + cần học docker/nginx (google)
- Vấn đề SEO ?

### **1 server chạy nhiều website ?**

Với công cụ là docker và nginx, sau khi bạn mua hosting, có thể chạy nhiều website (tên miền khác nhau) trên cùng 1 hosting (server) nhé: <https://superuser.com/a/1657705>

(cách làm là trong file config của nginx, định nghĩa thêm server là được, clone code cũ, sau đấy đổi tên của server name :v )

Monitoring ? xoá image docker ? docker system prune -af

## Chapter 9: Run Web App Free với Virtual Host (Localhost)

Mục đích mình làm chương này, để cho các bạn không có điều kiện mua tên miền & hosting, có thể test full mô hình của 1 website (thông qua virtual box)

### #78. Setup Docker với Virtual Box & Linux Alpine

Tài liệu:

<https://donotblock.me/https://www.c-sharpcorner.com/article/docker-installation-in-windows-system-through-oracle-virtual-box/>

<https://www.c-sharpcorner.com/article/docker-installation-in-windows-system-through-oracle-virtual-box/>

**Files sử dụng trong video:**

[https://drive.google.com/file/d/1bDd9ktpjn732XU\\_dPrT7IF6IFHb\\_kFb2/view?usp=share\\_link](https://drive.google.com/file/d/1bDd9ktpjn732XU_dPrT7IF6IFHb_kFb2/view?usp=share_link)

1. download and install virtual box:

<https://www.virtualbox.org/wiki/Downloads>

2. Download Alpine-Linux distribution:

=> download x86\_64 (standard)

<https://www.alpinelinux.org/downloads/>

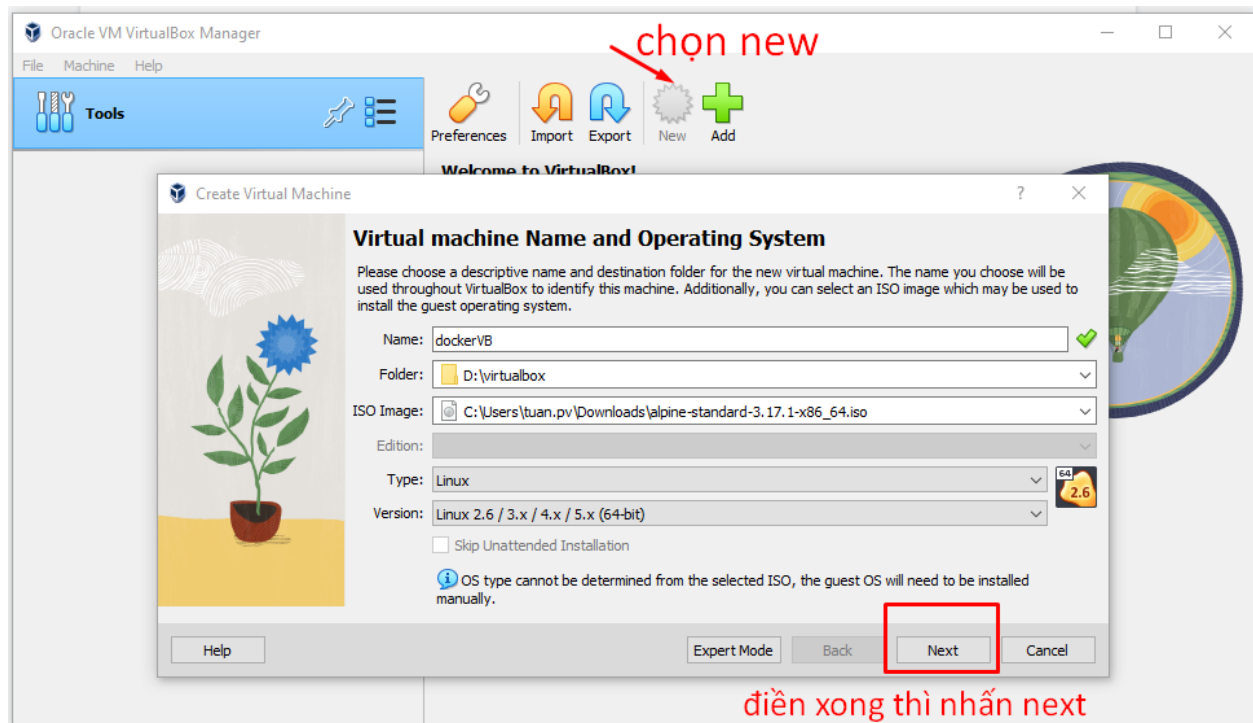
3. Install Linux with Virtual Box

Các bước làm:

#### **Bước 1: Tại giao diện VirtualBox => New (setup cho linux như ảnh)**

trong đó:

- Name: là tên của máy ảo (tự đặt tên tùy thích), mình họa với tên: dockerVB
- Folder: lưu trữ data của máy ảo (có thể chọn nơi lưu trữ tùy thích, hoặc để mặc định). mình dùng nơi lưu trữ custom (để giảm dung lượng cho ổ C), mình họa với tên: D:\virtualbox
- ISO: file linux image đã download tại bước 2 ở trên
- type: linux
- version: linux 2.6/3.x/4.x/5.x(64bit) => dùng cho máy 64bit, nếu máy bạn windows 32bit thì chọn 32bit



## Chọn Next

### Bước 2: Setup Hardware:

- Base Memory:

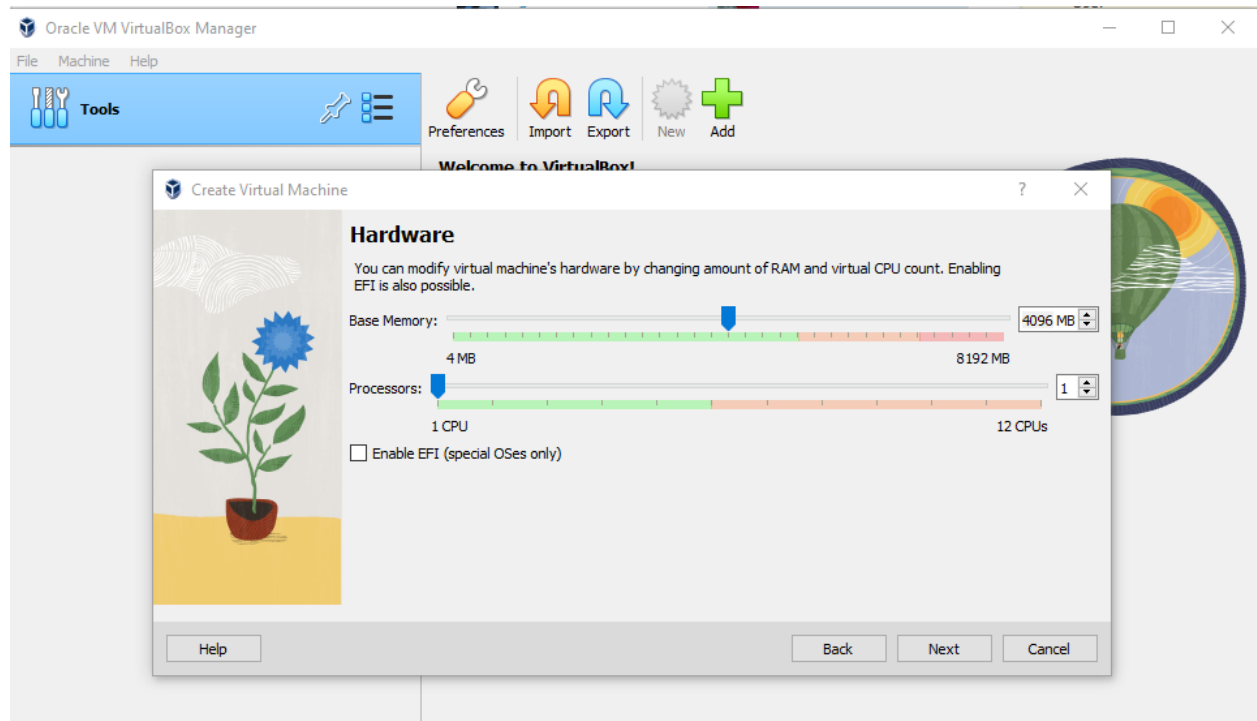
Docker recommend 4GB RAM (4096 MB)

Nếu máy bạn 8GB RAM => Setup 4GB

4GB => setup 2GB (case này sợ không chạy được)

nhều hơn 8GB RAM => Setup bao nhiêu tùy thích (vì càng nhiều tài nguyên => máy ảo càng mạnh)

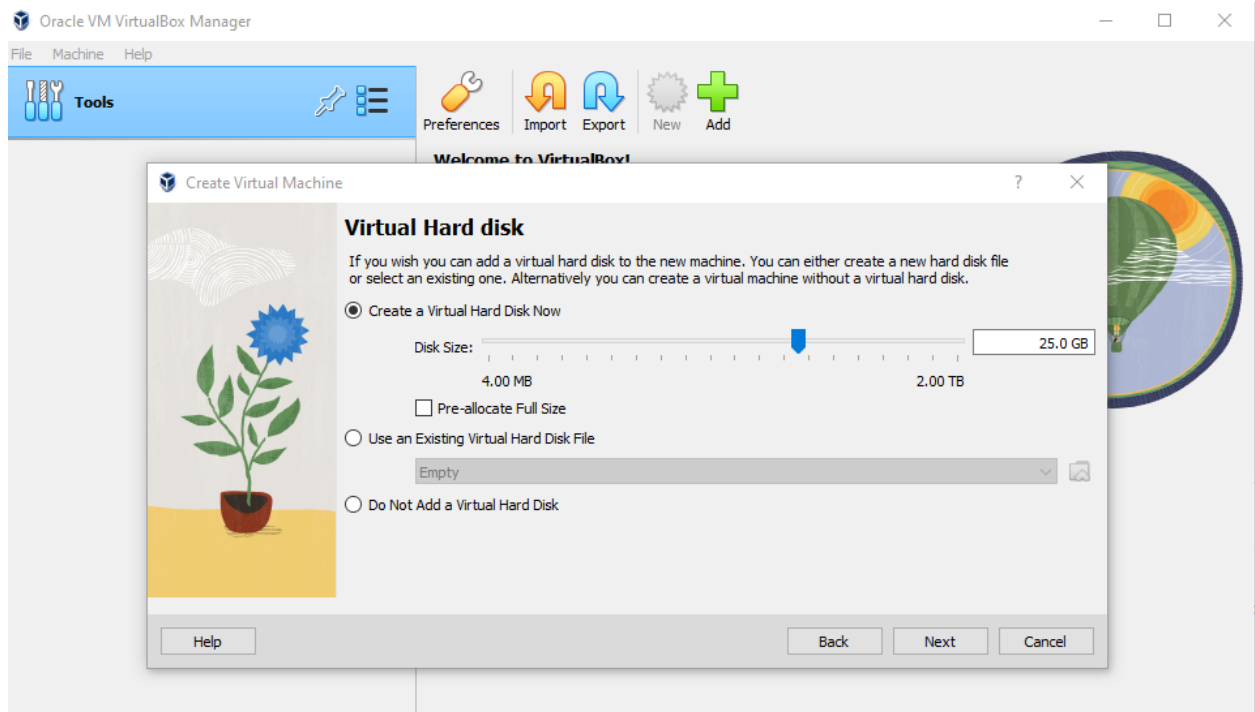
- Processors: để mặc định 1



**Chọn Next**

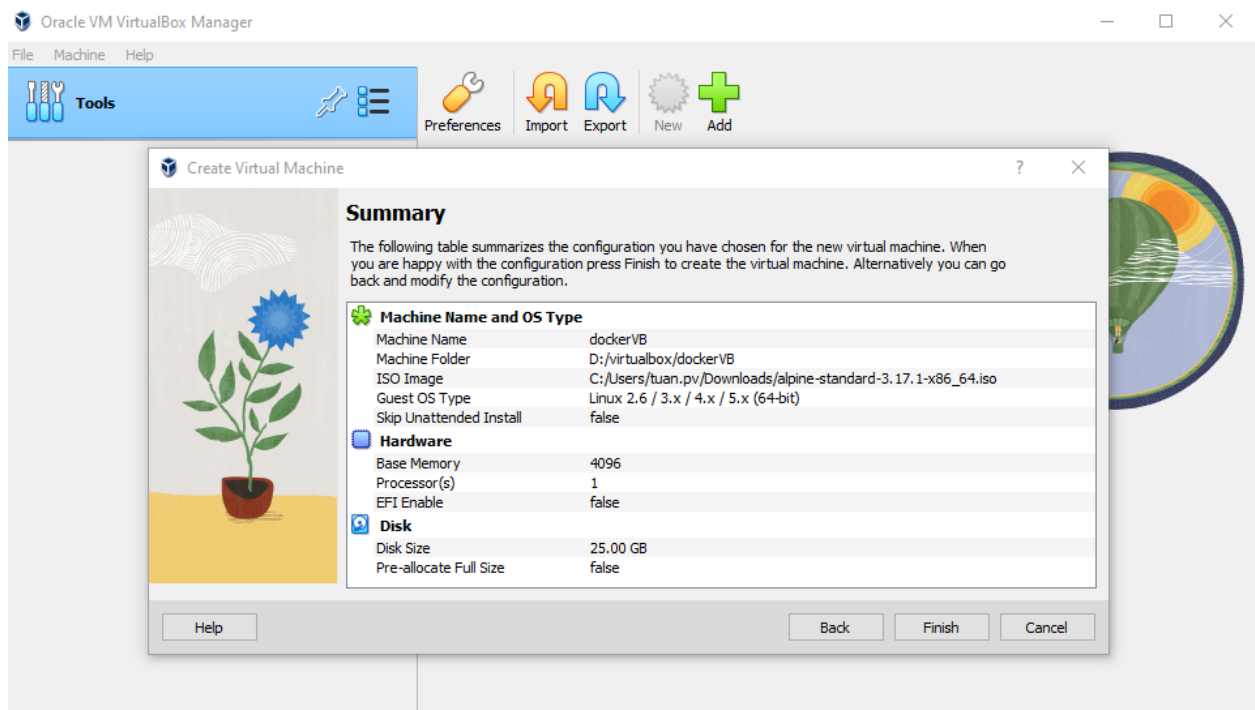
### **Bước 3: Setup Virtual Hard disk**

Nếu ổ đĩa bạn có nhiều dung lượng, thì nên chọn nhiều => ở đây, mình chọn 25GB



Chọn Next

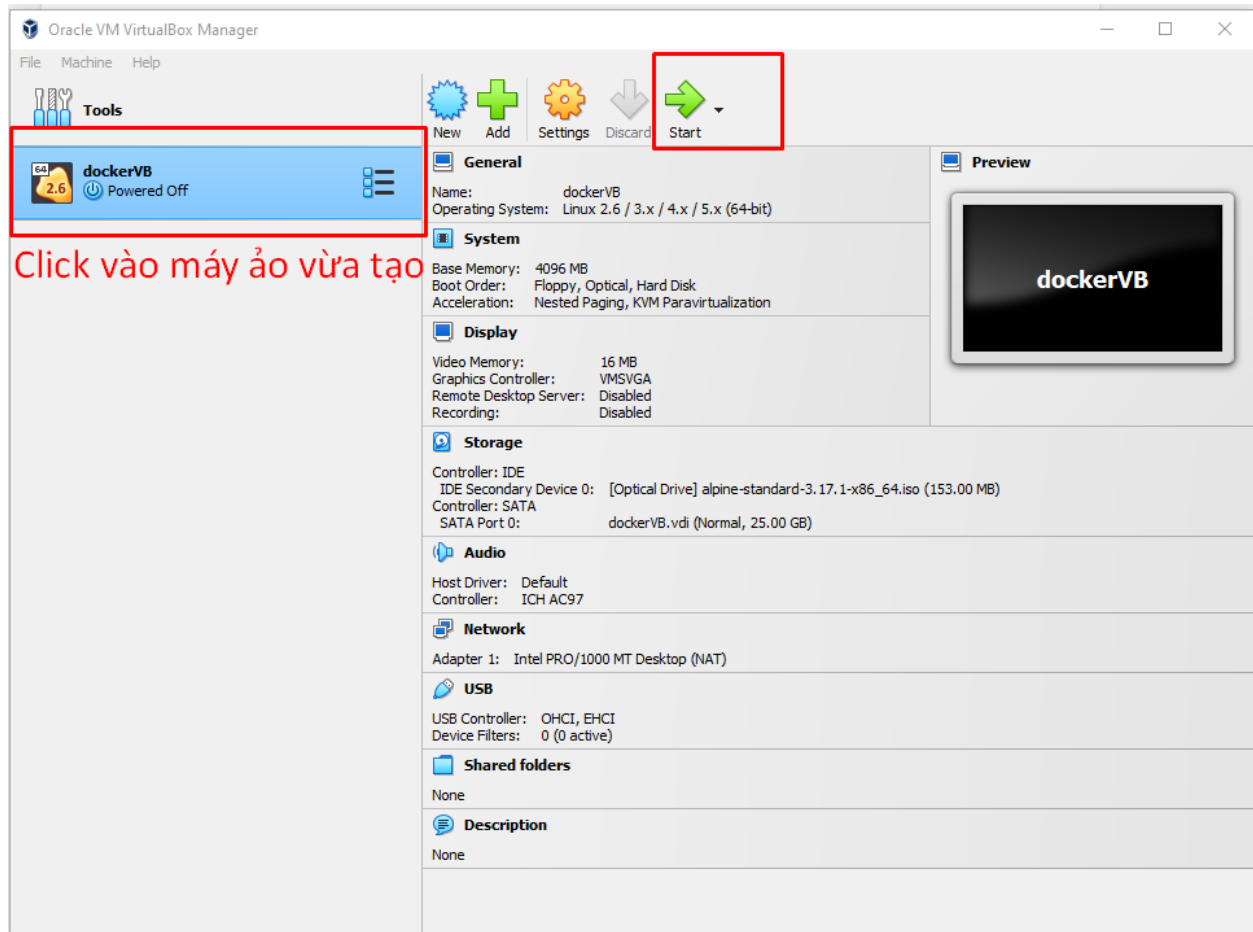
#### Bước 4: Review lại setup configuration



Chọn Finish



## Bước 5: Start máy ảo vừa tạo

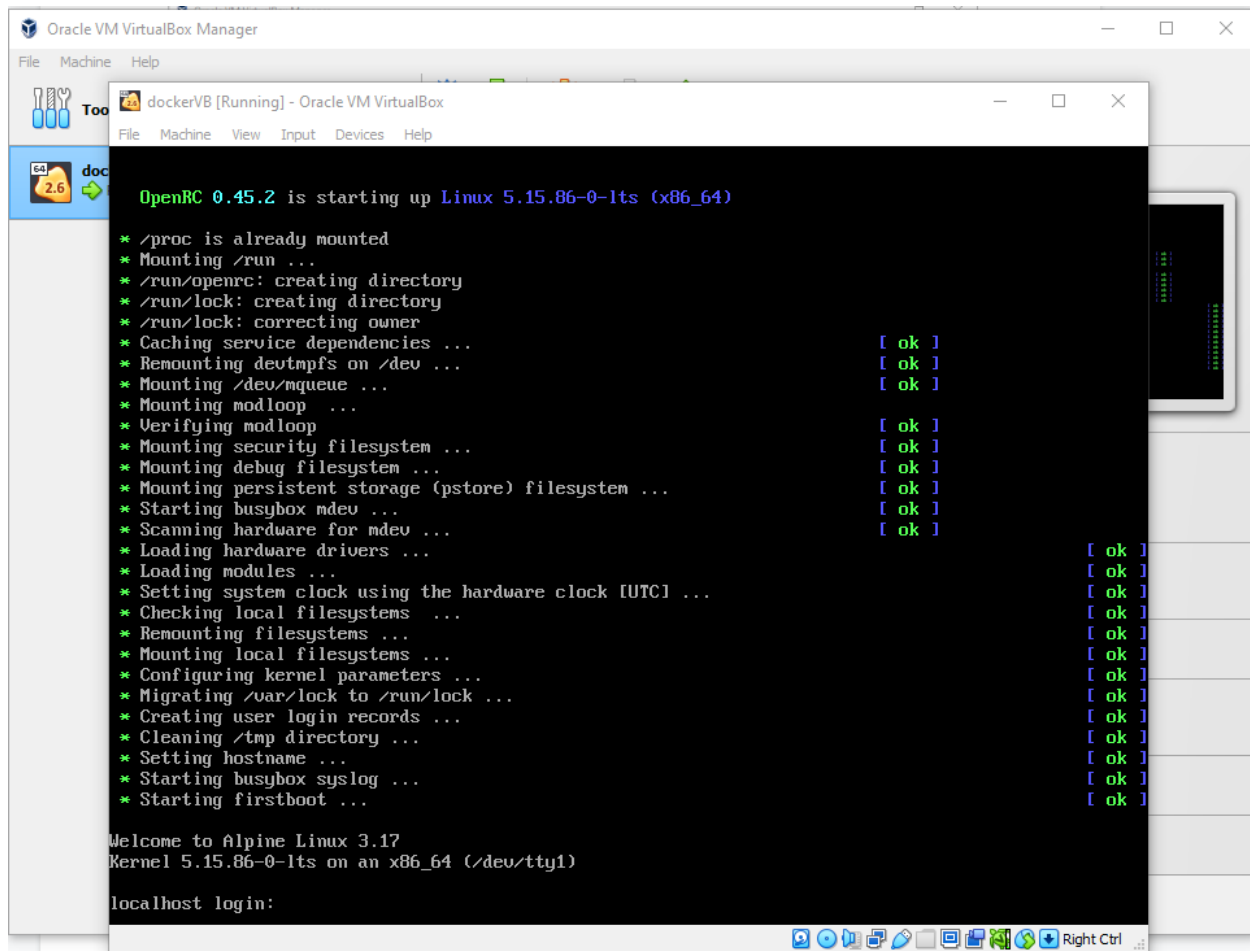


Để chạy máy ảo, có 2 cách:

Cách 1: Click vào máy ảo vừa tạo => nhấn nút Start như hình trên

Cách 2: Click vào máy ảo vừa tạo => Nhấn chuột phải => Start => Normal Start

Nếu chạy máy ảo thành công, sẽ có giao diện **tương tự** thế này:



## **Bước 6: Setup thông tin cho máy ảo vừa tạo**

Các thông tin setup:

localhost login: nhập vào root => nhấn enter

Tiếp theo gõ, setup-alpine

Keyboard setup:

- select keyboard layout: us
- select variant: us

System hostname (localhost) => nhấn enter lấy luôn giá trị mặc định

Available interfaces: eth0 => nhấn enter

Ip address for eth0 => nhấn enter

Do you want to do any manual network configuration => nhấn enter (nope)

changing password for root:

đây là môi trường test, mình khuyến khích đặt mật khẩu đơn giản => dễ nhớ, không nên phức tạp hóa vấn đề  
còn trong thực tế, các bạn lưu ý nên đặt mật khẩu đủ mạnh.

ở đây, với New password: mình nhập 123456 => nhấn enter. retype password làm tương tự

Which timezone are you in ? => enter

HTTP/FTP proxy URL ? => enter

Which NTP client to run => enter

nhấn enter

which ssh server => enter

allow root ssh login => yes

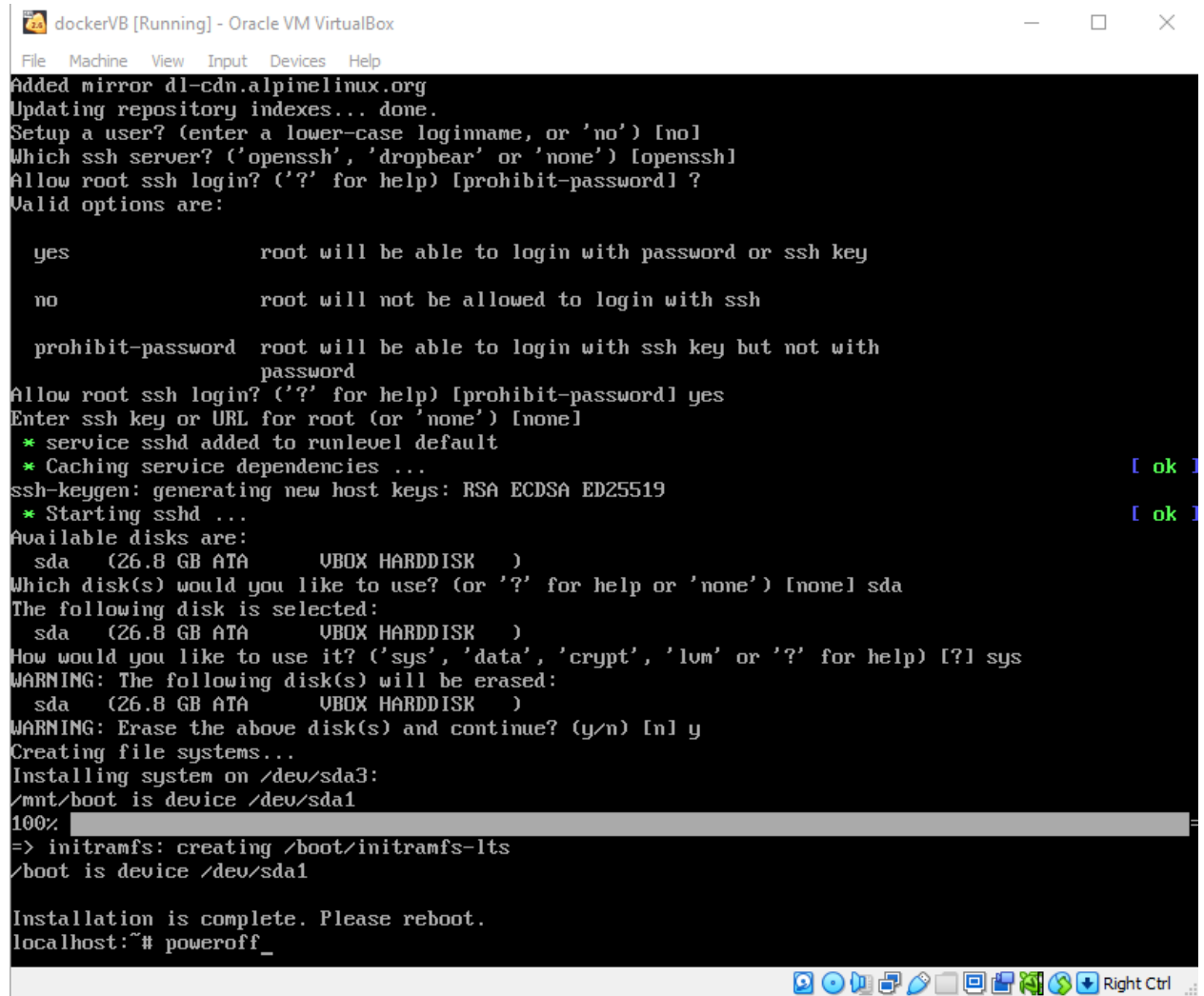
enter ssh key ... => enter

Which disk would you like to use ? => sda

how would you like to use it => sys

erase the above disk and continue => y

quá trình trên tốn từ 1 tới 5 phút . sau khi cài đặt, cần reboot  
=> gõ poweroff



```
dockerVB [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Added mirror dl-cdn.alpinelinux.org
Updating repository indexes... done.
Setup a user? (enter a lower-case loginname, or 'no') [no]
Which ssh server? ('openssh', 'dropbear' or 'none') [openssh]
Allow root ssh login? ('?' for help) [prohibit-password] ?
Valid options are:

yes                root will be able to login with password or ssh key

no                root will not be allowed to login with ssh

prohibit-password  root will be able to login with ssh key but not with
                    password
Allow root ssh login? ('?' for help) [prohibit-password] yes
Enter ssh key or URL for root (or 'none') [none]
* service sshd added to runlevel default
* Caching service dependencies ... [ ok ]
ssh-keygen: generating new host keys: RSA ECDSA ED25519
* Starting sshd ... [ ok ]
Available disks are:
sda (26.8 GB ATA VBOX HARDDISK )
Which disk(s) would you like to use? (or '?' for help or 'none') [none] sda
The following disk is selected:
sda (26.8 GB ATA VBOX HARDDISK )
How would you like to use it? ('sys', 'data', 'crypt', 'lvm' or '?' for help) [?] sys
WARNING: The following disk(s) will be erased:
sda (26.8 GB ATA VBOX HARDDISK )
WARNING: Erase the above disk(s) and continue? (y/n) [n] y
Creating file systems...
Installing system on /dev/sda3:
/mnt/boot is device /dev/sda1
100%
=> initramfs: creating /boot/initramfs-lts
/boot is device /dev/sda1

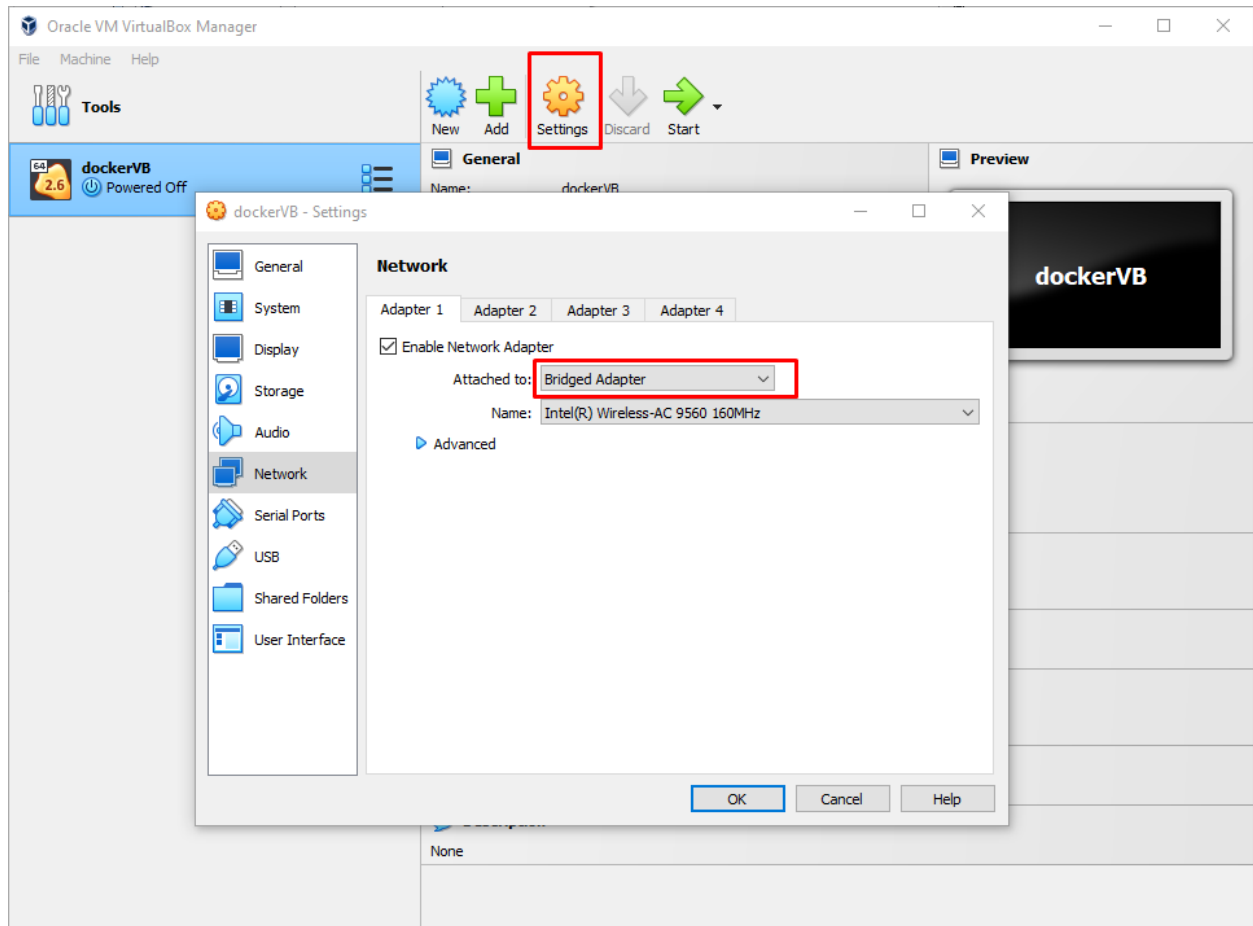
Installation is complete. Please reboot.
localhost:~# poweroff_
```

## Bước 7: Setup Docker

### Setup Network:

Chọn Settings => Chọn tab Network. Chọn Bridged Adapter

Lưu ý: phần Name để mặc định



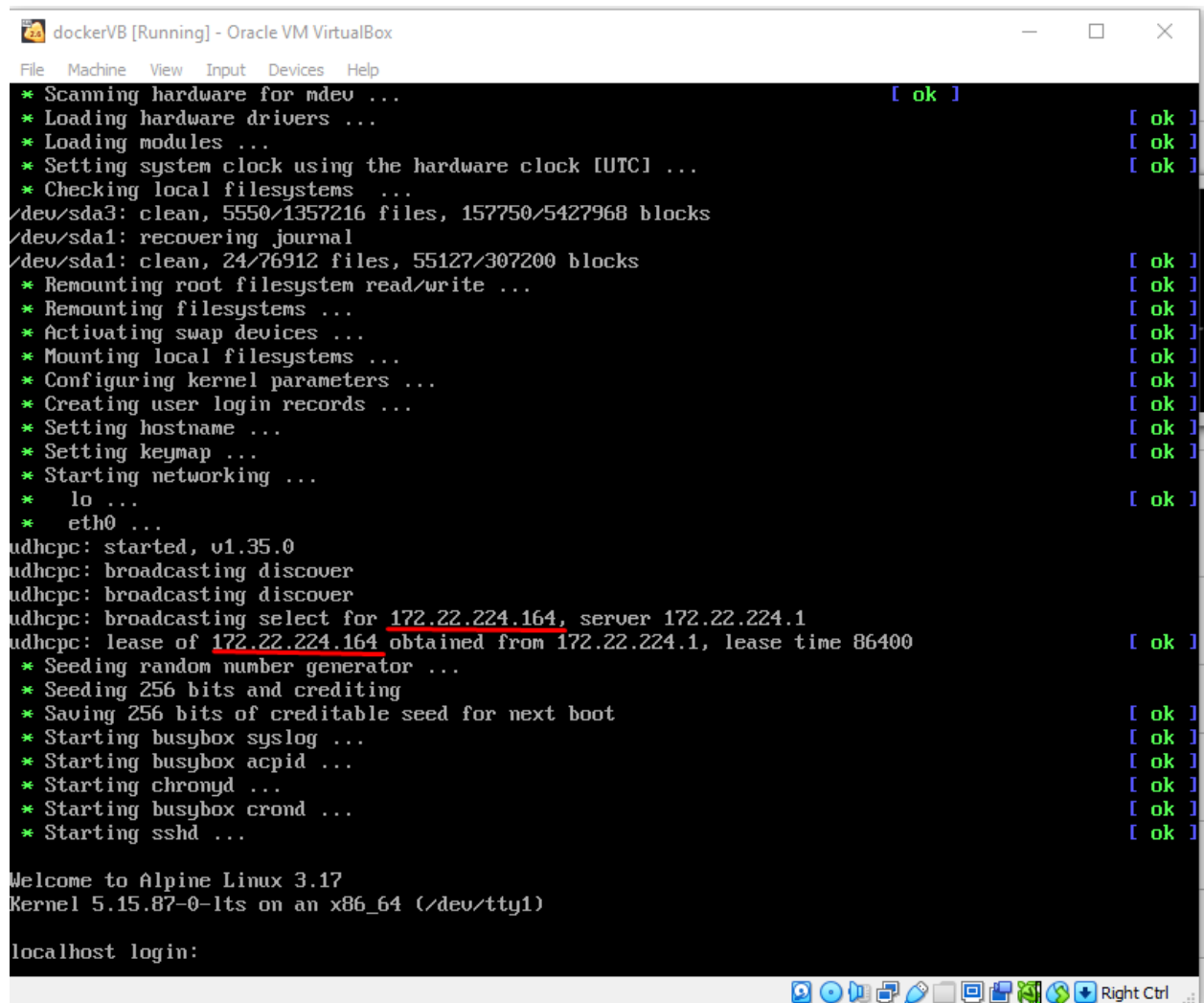
Tab System => lựa chọn như hình (chỉ giữ lại Hard Disk)

=> nhấn Ok để lưu cài đặt

**Nhấn Start để chạy máy ảo.**

(Nếu bị lỗi udhcpd broadcasting discover failed to get a DHCP lease virtual box  
=> change tab network

**Lần này, khi chạy, sẽ thấy được chỉ ip máy ảo.**



```
dockerVB [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
* Scanning hardware for mdev ... [ ok ]
* Loading hardware drivers ... [ ok ]
* Loading modules ... [ ok ]
* Setting system clock using the hardware clock [UTC] ... [ ok ]
* Checking local filesystems ...
/dev/sda3: clean, 5550/1357216 files, 157750/5427968 blocks
/dev/sda1: recovering journal
/dev/sda1: clean, 24/76912 files, 55127/307200 blocks [ ok ]
* Remounting root filesystem read/write ... [ ok ]
* Remounting filesystems ... [ ok ]
* Activating swap devices ... [ ok ]
* Mounting local filesystems ... [ ok ]
* Configuring kernel parameters ... [ ok ]
* Creating user login records ... [ ok ]
* Setting hostname ... [ ok ]
* Setting keymap ... [ ok ]
* Starting networking ... [ ok ]
*   lo ... [ ok ]
*   eth0 ...
udhcpd: started, v1.35.0
udhcpd: broadcasting discover
udhcpd: broadcasting discover
udhcpd: broadcasting select for 172.22.224.164, server 172.22.224.1
udhcpd: lease of 172.22.224.164 obtained from 172.22.224.1, lease time 86400 [ ok ]
* Seeding random number generator ...
* Seeding 256 bits and crediting
* Saving 256 bits of creditable seed for next boot [ ok ]
* Starting busybox syslog ... [ ok ]
* Starting busybox acpid ... [ ok ]
* Starting chronyd ... [ ok ]
* Starting busybox crond ... [ ok ]
* Starting sshd ... [ ok ]

Welcome to Alpine Linux 3.17
Kernel 5.15.87-0-lts on an x86_64 (/dev/tty1)

localhost login:
```

lưu thông tin này lại, như trên hình, là 172.22.224.164

lưu ý: đây là ip máy mình, máy bạn sẽ khác =))

login với thông tin user root/123456 (nếu bạn ko đặt pass là 123456 thì thay vào tương ứng)

Cần edit file sau:

gõ:

**vi /etc/apk/repositories**

nhấn phím "i"

bỏ tất cả các dấu # => nhấn phím ESC => nhấn phím :wq

- Cài đặt docker và docker compose với câu lệnh:

apk add docker docker-cli-compose

--verify:

docker version

docker compose version

chạy docker với câu lệnh: service docker start

Kiểm tra docker đã chạy được chưa:

docker run -p 8888:80 -d nginx

để check, truy cập vào: địa chỉ ip máy bạn (đã tạo ở các bước ở trên), sau đấy là port

ví dụ: <http://172.22.224.164:8888/>

Lưu ý: mỗi lần tắt máy, nên lưu lại state của máy ảo => lần sau mở lên, nó sẽ có y nguyên như lúc đã tắt

Nếu docker chưa chạy, sử dụng: **service docker start**

Bonus: (run all container if not running): **docker restart \$(docker ps -q)**

## #79. Copy Files Từ Windows Host Vào Virtual Machine Linux

Share files from windows host to virtual box: (mount)

**Bước 1: Tạo folder (với content) muốn share tại windows**

**Bước 2: set up share folder, trong đó:**

- Click chọn Virtual Machine => chọn Settings => chọn tab Shared Folders => add news

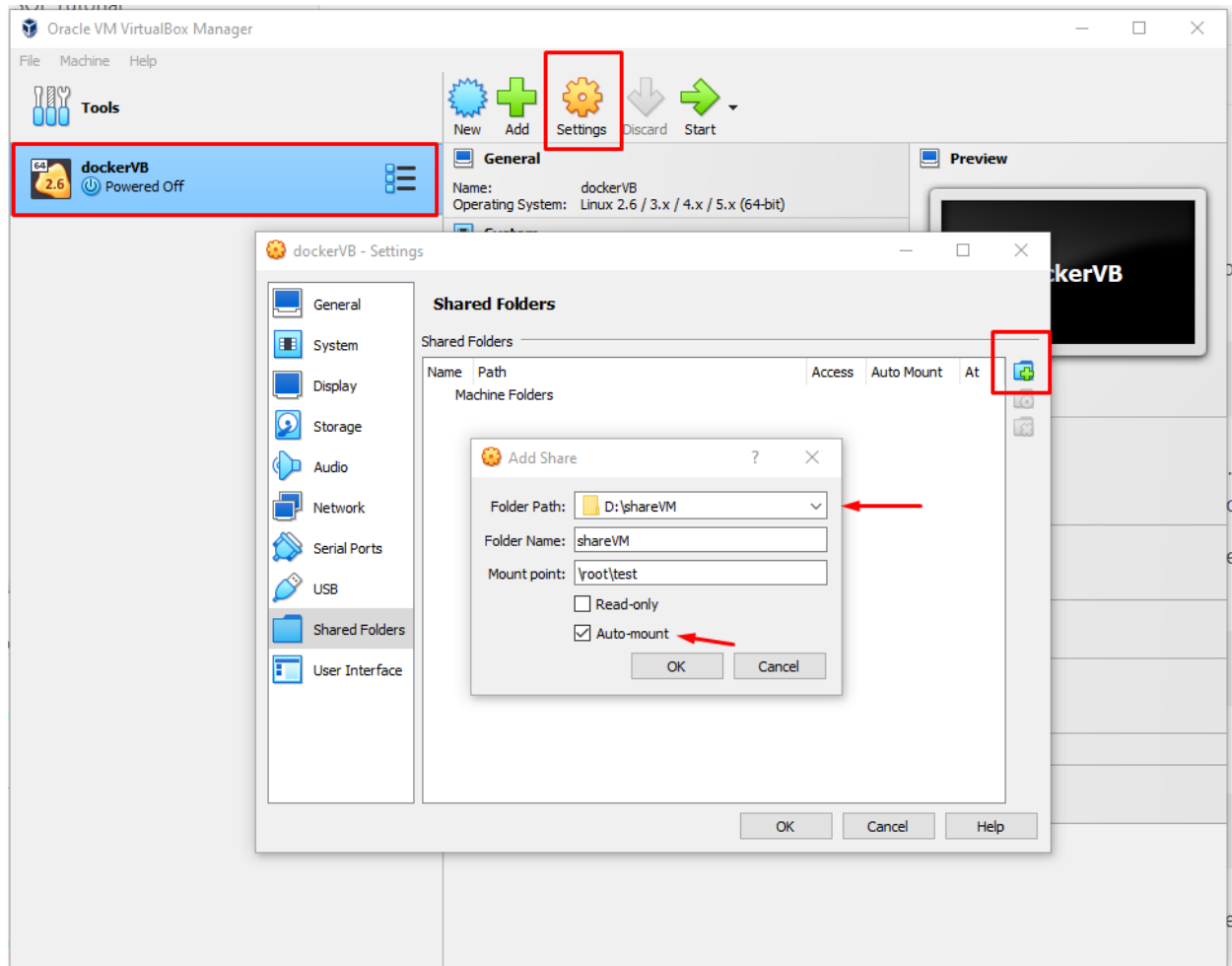
+ folder path: chọn thư mục bạn muốn share từ máy windows (host) vào máy ảo (virtual machine)

+ folder name sẽ được tự động filled.

+ Mount point (nơi lưu trữ trên máy ảo), có thể ghi hoặc không

+ chọn tick auto-mount





### Bước 3: start máy ảo như bình thường

mount data với câu lệnh:

```
mount -t vboxsf from_host to_virtual
```

ví dụ:

```
mount -t vboxsf shareVM /root/final321
```

```
mount -t vboxsf sharingVM /root/hoidanit
```

```
dockerReact /root/eric
```

bonus: copy content sharing:

- tạo folder mới

`mkdir folder_name`

- copy all files from A to B

`cp -a source/ . destination/`

(đứng tại current directory)

read file: vi file\_name

**`docker compose -f mysql.yml -p hoidanit-sql up -d`**

## **Chapter 10: Build App với Next.JS và Nginx**

### **#80. Deploy Hosting FREE cho Next.JS**

Tương tự như deploy ứng dụng React dùng với Single Page App ( như Vite, CRA...), bạn có thể dùng Vercel để triển khai Next.js hoàn toàn miễn phí.

Trang chủ: <https://vercel.com/>

Các bước cần làm:

1. Tạo tài khoản Vercel
2. Liên kết github tới project trên Vercel của bạn
3. Mỗi lần commit code lên github, Vercel sẽ tự động build lại dự án cho bạn.  
Amazing :v

Tuy nhiên, cần lưu ý là, Next.js hay Vercel, chỉ làm frontend thôi nhé. Thực tế là vậy đấy.

Đừng dùng Vercel làm backend cho bạn, vì:

1. Làm nặng ứng dụng Frontend (cụ thể là Nextjs), vì khi này, bạn “viết thêm package trong Next.js” để làm backend.
2. Vercel không hỗ trợ việc lưu trữ database, vì vậy, bạn cần lưu trữ database ở nơi khác

## **#81.Build Next.JS với Docker**

Tài liệu: <https://nextjs.org/docs/app/building-your-application/deploying#self-hosting>

Nội dung file Docker tham khảo:

<https://drive.google.com/file/d/1IK5rWPFUgscqql9BLMIGge2hJkG0sGQF/view?usp=sharing>

## #82. Lưu Ý Khi Build Docker với Next.JS

### 1. URL Backend cho next.js

Thông thường, bạn **chạy Next.JS tại: localhost:3000**

**Và chạy backend tại: localhost:8000**

Các api được sử dụng, ví dụ, api login:  
**localhost:8000/api/...**

**Khi dùng Nginx, bạn điều hướng url “/api” sang backend**

**=> khi build docker, bạn cần set URL\_BACKEND = /**

Bởi vì bây giờ, bạn chạy nextjs trên url xuất phát là “/”, bạn gọi api với url “/api/...”

### 2. Images cho next.js

**Cần check file next.config.js**

**<https://nextjs.org/docs/pages/api-reference/next-config-js/output#automatically-copying-traced-files>**

### **#83. Build NextJS với Docker Compose**

#### **Lưu ý:**

Với Server yếu, không nên build trực tiếp NextJS trên đây.

Thay vì vậy, bạn có thể “build trực tiếp trên máy tính cá nhân”. Lên hosting VPS, chỉ cần kéo image về dùng thôi

### **#84. Build NextJS với Nginx**

Todo...

## Lời Kết

Như vậy là chúng ta đã cùng nhau trải qua hơn 80+ videos về học & hiểu cách triển khai một ứng dụng website thực tế lên môi trường production (với công nghệ React/Node.JS).

Với công nghệ mình sử dụng trong khóa học (docker/nginx), sau này, chỉ cần bạn viết được docker file & chạy tại local, chắc chắn có thể triển khai được ứng dụng chạy thực tế.

Dĩ nhiên rằng, trong quá trình quá trình thực hiện khóa học này, mình sẽ không thể tránh khỏi những sai sót,

Vì vậy, nếu thấy sai sót, các bạn cứ thoải mái đóng góp qua Fanpage Hỏi Dân IT nhé.

<https://www.facebook.com/askITwithERIC>

Với khóa học này, mình hy vọng nó sẽ là hành trang để giúp bạn có thêm kiến thức trên con đường IT của bản thân, nhanh có nhiều 'năng lượng' cho công việc. (năng lượng === lương nặng)

Sau khóa học này, sẽ là update và thêm mới các khóa học...

Hẹn gặp lại các bạn ở các khóa học tiếp theo ....

**Hỏi Dân IT**

