

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**HCMUTE**

**ĐỒ ÁN MÔN HỌC**  
**ĐIỆN TOÁN ĐÁM MÂY**

**ĐỀ TÀI**  
**XÂY DỰNG ỨNG DỤNG ĐƠN GIẢN VÀ**  
**DEPLOY LÊN KUBERNETES**

**SVTH:**

**Nguyễn Huỳnh Phúc**

**Phạm Văn Minh Tân**

**GVHD: TS. Huỳnh Xuân Phụng**

**MSSV:**

**17110350**

**17110364**

**Tp. Hồ Chí Minh, tháng 1 năm 2021**

## MỤC LỤC

<b>BẢNG PHÂN CÔNG NHIỆM VỤ, CHỨC NĂNG .....</b>	<b>4</b>
<b>CHƯƠNG 1: GIỚI THIỆU CHUNG .....</b>	<b>5</b>
1.1. Lý do chọn đề tài .....	5
1.2. Mục tiêu của đề tài .....	5
1.3. Bố cục của báo cáo .....	5
<b>CHƯƠNG 2: KHÁI NIỆM.....</b>	<b>6</b>
2.1. Điện toán đám mây .....	6
2.2. Docker và container trong docker.....	6
2.2.1. Docker .....	6
2.2.2. Container trong docker .....	6
2.3. Kubernetes .....	7
<b>CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG VÀ DEPLOY LÊN KUBERNETES.....</b>	<b>9</b>
3.1. Công nghệ và phần mềm sử dụng .....	9
3.2. Xây dựng ứng dụng.....	9
3.3. Deploy ứng dụng lên Kubernetes .....	13
3.3.1. Xây dựng docker image cho back-end.....	13
3.3.2. Xây dựng docker image cho front-end .....	14
3.3.3. Deploy ứng dụng lên Kubernetes .....	16
<b>CHƯƠNG 4: KẾT LUẬN .....</b>	<b>20</b>
4.1. Kết quả đạt được.....	20
4.2. Các hạn chế.....	20

<b>4.3. Hướng phát triển.....</b>	<b>20</b>
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>21</b>

## **BẢNG PHÂN CÔNG NHIỆM VỤ, CHỨC NĂNG**

STT	Công việc	Người thực hiện	Đánh giá
1	Phân tích nghiệp vụ	Phạm Văn Minh Tân	Hoàn thành
2	Lập trình front-end	Nguyễn Huỳnh Phúc	Hoàn thành
3	Lập trình back-end	Phạm Văn Minh Tân	Hoàn thành
4	Deploy lên Kubernetes	Nguyễn Huỳnh Phúc	Hoàn thành

## **CHƯƠNG 1: GIỚI THIỆU CHUNG**

### **1.1. Lý do chọn đề tài**

Từ nhu cầu thực tiễn, các tổ chức, doanh nghiệp và cá nhân luôn muốn hạn chế chi phí cho các tài nguyên công nghệ thông tin, đặc biệt là các máy chủ. Điện toán đám mây là giải pháp để giải quyết vấn đề trên. Rất nhiều dịch vụ điện toán đám mây yêu cầu nền tảng dựa trên Kubernetes (Kubernetes-based platform).

Nhóm quyết định tìm hiểu về Kubernetes với đề tài: **“Xây dựng ứng dụng đơn giản và deploy lên Kubernetes”**.

### **1.2. Mục tiêu của đề tài**

- Xây dựng ứng dụng: trang web sinh ngẫu nhiên đơn giản
- Deploy ứng dụng đã xây dựng lên Kubernetes

### **1.3. Bố cục của báo cáo**

Chương 1: Giới thiệu chung

Chương 2: Khái niệm

Chương 3: Xây dựng ứng dụng và deploy lên kubernetes

Chương 4: Kết luận

## **CHƯƠNG 2: KHÁI NIỆM**

### **2.1. Điện toán đám mây**

Điện toán đám mây (tiếng Anh: cloud computing), còn gọi là điện toán máy chủ ảo, là mô hình điện toán sử dụng các công nghệ máy tính và phát triển dựa vào mạng Internet. Thuật ngữ "đám mây" ở đây là lối nói ẩn dụ chỉ mạng Internet (dựa vào cách được bố trí của nó trong sơ đồ mạng máy tính) và như sự liên tưởng về độ phức tạp của các cơ sở hạ tầng chứa trong nó. Ở mô hình điện toán này, mọi khả năng liên quan đến công nghệ thông tin đều được cung cấp dưới dạng các "dịch vụ", cho phép người sử dụng truy cập các dịch vụ công nghệ từ một nhà cung cấp nào đó "trong đám mây" mà không cần phải có các kiến thức, kinh nghiệm về công nghệ đó, cũng như không cần quan tâm đến các cơ sở hạ tầng phục vụ công nghệ đó. Theo tổ chức IEEE "Nó là hình mẫu trong đó thông tin được lưu trữ thường trực tại các máy chủ trên Internet và chỉ được được lưu trữ tạm thời ở các máy khách, bao gồm máy tính cá nhân, trung tâm giải trí, máy tính trong doanh nghiệp, các phương tiện máy tính cầm tay,...". Điện toán đám mây là khái niệm tổng thể bao gồm cả các khái niệm như phần mềm dịch vụ, Web 2.0 và các vấn đề khác xuất hiện gần đây, các xu hướng công nghệ nổi bật, trong đó đề tài chủ yếu của nó là vấn đề dựa vào Internet để đáp ứng những nhu cầu điện toán của người dùng.

### **2.2. Docker và container trong docker**

#### **2.2.1. Docker**

Docker là một nền tảng để cung cấp cách để building, deploying và running ứng dụng dễ dàng hơn bằng cách sử dụng các containers (trên nền tảng ảo hóa). Ban đầu viết bằng Python, hiện tại đã chuyển sang Golang.

#### **2.2.2. Container trong docker**

Các containers cho phép lập trình viên đóng gói một ứng dụng với tất cả các phần cần thiết, chẳng hạn như thư viện và các phụ thuộc khác, và gói tất cả ra dưới dạng một package.

Bằng cách đó, nhờ vào container, ứng dụng sẽ chạy trên mọi máy Linux khác bất kể mọi cài đặt tùy chỉnh mà máy có thể có khác với máy được sử dụng để viết code.

Theo một cách nào đó, Docker khá giống virtual machine. Nhưng tại sao Docker lại phát triển, phổ biến nhanh chóng? Đây là những nguyên nhân:

- Tính dễ ứng dụng: Docker rất dễ cho mọi người sử dụng từ lập trình viên, sys admin... nó tận dụng lợi thế của container để build, test nhanh chóng. Có thể đóng gói ứng dụng trên laptop của họ và chạy trên public cloud, private cloud... Câu thần chú là “Build once, run anywhere”.

- Tốc độ: Docker container rất nhẹ và nhanh, bạn có thể tạo và chạy docker container trong vài giây.

- Môi trường chạy và khả năng mở rộng: Bạn có thể chia nhỏ những chức năng của ứng dụng thành các container riêng lẻ. Ví dụ Database chạy trên một container và Redis cache có thể chạy trên một container khác trong khi ứng dụng Node.js lại chạy trên một cái khác nữa. Với Docker, rất dễ để liên kết các container với nhau để tạo thành một ứng dụng, làm cho nó dễ dàng scale, update các thành phần độc lập với nhau.

Với xu hướng dịch chuyển sang microservices của các hệ thống lớn, Docker đang làm một thành phần cực kỳ quan trọng, làm cho nó trở thành một phần của nhiều công cụ DevOps. Hiện tại thế giới bắt đầu sử dụng thêm một công cụ quản lý container tiên tiến khác là Kubernetes.

### **2.3. Kubernetes**

Kubernetes, hoặc k8s là một nền tảng mã nguồn mở tự động hoá việc quản lý, scaling và triển khai ứng dụng dưới dạng container hay còn gọi là Container orchestration engine. Nó loại bỏ rất nhiều các quy trình thủ công liên quan đến việc triển khai và mở rộng các containerized applications.

Kubernetes orchestration cho phép bạn xây dựng các dịch vụ ứng dụng mở rộng nhiều containers. Nó lên lịch các containers đó trên một cụm, mở rộng các containers và quản lý tình trạng của các containers theo thời gian.

Các ứng dụng production thực tế mở rộng nhiều containers. Các containers đó phải được triển khai trên nhiều server hosts. Kubernetes cung cấp khả năng phối hợp và quản lý cần thiết để triển khai các containers theo quy mô cho các workloads đó.

Kubernetes ban đầu được phát triển và thiết kế bởi các kỹ sư tại Google. Đây cũng là công nghệ đằng sau các dịch vụ đám mây của Google. Google đã và đang tạo ra hơn 2 tỷ container deployments mỗi tuần và tất cả đều được hỗ trợ bởi nền tảng nội bộ: Borg.



## CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG VÀ DEPLOY LÊN KUBERNETES

### 3.1. Công nghệ và phần mềm sử dụng

- Visual studio 2019 v16.6.5
- Visual studio Code 1.52.1
- ASP.NET Core 3.1
- Angular 10
- Docker Desktop 2.4.0.0
- Kubernetes v1.18.8

### 3.2. Xây dựng ứng dụng

#### *Layout*

- Tiêu đề trang web
- Menu chức năng
- Nội dung chức năng
- Footer trang web

KubedemoApp x +

← → ↻ 📄 🌟 📄 📄 📄 📄 📄

localhost:4100

## Random.tanphuc

Integer List Randomizer Decimal Strings

Min value:

Max value:

Result:

Copyright © - HCMC University of Technology and Education - Cloud Computing - Kubedemo

*Chức năng sinh ngẫu nhiên số nguyên trong khoảng nhập vào*

- Giao diện: nhãn, trường nhập giá trị khoảng, nút sinh ngẫu nhiên, nhãn kết quả và nhãn hiển thị kết quả

<u>Integer</u>	List Randomizer	Decimal	Strings
Min value:	<input type="text" value="1"/>		
Max value:	<input type="text" value="100"/>		
<input type="button" value="Generate random number"/>			
Result:	<b>81</b>		

- Cài đặt xử lí

```
[HttpGet("RandomInteger/{minValue=0}/{maxValue=int.MaxValue}")]
0 references
public async Task<IActionResult> RandomInteger(int minValue = 0, int maxValue = int.MaxValue)
{
    return new ObjectResult(await Task.Run(() => random.Next(minValue, maxValue)));
}
```

Chức năng sinh danh sách mới với thứ tự ngẫu nhiên từ danh sách nhập vào

- Giao diện: nhấn, trường nhập danh sách, nút sinh ngẫu nhiên, nhấn kết quả và nhấn hiển thị kết quả

Integer	<u>List Randomizer</u>	Decimal	Strings
List value:	<input type="text" value="                 Tân                 Phúc                 Tân Phúc                 Phúc Tân"/>		
<input type="button" value="Generate random from list"/>			
Result:	<b>Phúc Tân</b> <b>Phúc</b> <b>Tân Phúc</b> <b>Tân</b>		

- Cài đặt xử lí

```
[HttpGet("ListRandomizer")]
0 references
public async Task<IActionResult> ListRandomizer(string input)
{
    return await Task.Run(() =>
    {
        string[] res = input.Split(Environment.NewLine.ToCharArray());
        int loop = random.Next(res.Length, 2 * res.Length);
        while (loop-- > 0)
        {
            int i = random.Next(res.Length);
            int j = random.Next(res.Length);
            (res[i], res[j]) = (res[j], res[i]);
        }
        return new ObjectResult(res);
    });
}
```

*Chức năng sinh số thực ngẫu nhiên*

- Giao diện: nút sinh ngẫu nhiên, nhấn kết quả và nhấn hiển thị kết quả

Integer	List Randomizer	<u>Decimal</u>	Strings
---------	-----------------	----------------	---------

Generate random decimal number

Result: **0.8763779284788192**

- Cài đặt xử lí

```
[HttpGet("RandomDecimal")]
0 references
public async Task<IActionResult> RandomDecimal()
{
    return new ObjectResult(await Task.Run(() => random.NextDouble()));
}
```

*Chức năng danh sách ngẫu nhiên (mỗi phần tử của danh sách là một chuỗi các ký tự)*

- Giao diện: nhấn, trường nhập số lượng phần tử và ký tự mỗi phần tử, nhấn và checkbox các lựa chọn cho phép chứa chữ số, ký tự in thường, ký tự in hoa, cho phép các phần tử giống nhau, nút sinh ngẫu nhiên, nhấn kết quả và nhấn hiển thị kết quả

Integer	List Randomizer	Decimal	<u>Strings</u>
---------	-----------------	---------	----------------

String count:

Character by string count:

- Options:
- ☒ Include digits
  - ☒ Include upper case
  - ☒ Include lower case
  - ☒ Allow duplicate strings

Generate random number

Result:

**ZbJpQZeVvu**  
**0rHjfAybh0**  
**r8YWrt4qBJ**  
**1iqa1hQTRE**  
**T569M03mMU**

- Cài đặt xử lí

```
[HttpGet("StringGenerator")]
References
public async Task<ActionResult> StringGenerator(int stringCount = 10, int characterCount = 10, bool digits = true,
    bool upperLetter = false, bool lowerLetter = false, bool allowDuplicate = false)
{
    return await Task.Run(() =>
    {
        string resources = "";
        if (digits)
            resources += "0123456789";
        if (upperLetter)
            resources += "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        if (lowerLetter)
            resources += "abcdefghijklmnopqrstuvwxyz";
        if (string.IsNullOrEmpty(resources))
            return new ObjectResult(new string[0]);

        List<string> res = new List<string>();
        while (res.Count < stringCount)
        {
            string s = "";
            string source = resources;
            while (s.Length < characterCount)
            {
                int t = random.Next(source.Length);
                s += source[t];
            }
            if (!allowDuplicate && res.Contains(s))
                continue;
            res.Add(s);
        }

        return new ObjectResult(res);
    });
}
```

### 3.3. Deploy ứng dụng lên Kubernetes

#### 3.3.1. Xây dựng docker image cho back-end

- Xây dựng Dockerfile:

- + “Kube1.CloudProject.API” là tên project
- + “Kube1.CloudProject.API/Kube1.CloudProject.API.csproj” là đường dẫn tới file project ASP.NET Core từ vị trí Dockerfile
- + “Kube1.CloudProject.API/” là thư mục được tạo
- + “Kube1.CloudProject.API.dll” là điểm khởi chạy ứng dụng

```
Dockerfile X
back-end > Dockerfile
1 #See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build your images
2
3 FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim AS base
4 WORKDIR /app
5 EXPOSE 80
6 EXPOSE 443
7 ARG environment=Development
8 ENV ASPNETCORE_ENVIRONMENT=$environment
9
10 FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build
11 WORKDIR /src
12 COPY ["Kube1.CloudProject.API/Kube1.CloudProject.API.csproj", "Kube1.CloudProject.API/"]
13 RUN dotnet restore "Kube1.CloudProject.API/Kube1.CloudProject.API.csproj"
14 COPY . .
15 WORKDIR "/src/Kube1.CloudProject.API"
16 RUN dotnet build "Kube1.CloudProject.API.csproj" -c Release -o /app/build
17
18 FROM build AS publish
19 RUN dotnet publish "Kube1.CloudProject.API.csproj" -c Release -o /app/publish
20
21 FROM base AS final
22 WORKDIR /app
23 COPY --from=publish /app/publish .
24 ENTRYPOINT ["dotnet", "Kube1.CloudProject.API.dll"]
```

- Xây dựng docker image

+ Từ vị trí Dockerfile chạy lệnh:

**docker build -t kubebackend:1.0 .** (có dấu chấm cuối)

+ Kết quả

```
[+] Building 7.9s (18/18) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 942B 0.0s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 382B 0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/core/sdk:3.1-buster 0.8s
=> [internal] load metadata for mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim 0.0s
=> [build 1/7] FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster@sha256:2b0b6e9b4ceda402eebce563fb4eb155689dc989 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 11.29kB 0.1s
=> [base 1/2] FROM mcr.microsoft.com/dotnet/core/aspnet:3.1-buster-slim 0.0s
=> CACHED [build 2/7] WORKDIR /src 0.0s
=> CACHED [build 3/7] COPY [Kube1.CloudProject.API/Kube1.CloudProject.API.csproj, Kube1.CloudProject.API/] 0.0s
=> CACHED [build 4/7] RUN dotnet restore "Kube1.CloudProject.API/Kube1.CloudProject.API.csproj" 0.0s
=> [build 5/7] COPY . . 0.1s
=> [build 6/7] WORKDIR /src/Kube1.CloudProject.API 0.1s
=> [build 7/7] RUN dotnet build "Kube1.CloudProject.API.csproj" -c Release -o /app/build 4.0s
=> [publish 1/1] RUN dotnet publish "Kube1.CloudProject.API.csproj" -c Release -o /app/publish 2.5s
=> CACHED [base 2/2] WORKDIR /app 0.0s
=> CACHED [final 1/2] WORKDIR /app 0.0s
=> CACHED [final 2/2] COPY --from=publish /app/publish . 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => writing image sha256:1ae0cda6437985f60694cdde7911df26a26c601c3639830dd5a5973fba2b1b92 0.0s
=> naming to docker.io/library/kubebackend:1.0 0.0s
```

+ Docker image được tạo trên Docker Desktop

kubebackend	1.0	1ae0cda64379	about 1 hour ago	207.7 MB
-------------	-----	--------------	------------------	----------

### 3.3.2. Xây dựng docker image cho front-end

- Cài đặt api endpoint cho môi trường production: cổng kết nối 4100 là cổng kết nối công khai ra bên ngoài của cụm được deploy, sẽ được trình bày khi xây dựng file deploy.yaml

```
if (environment.production) {  
  this.domain = 'kubedemo.hcmute.edu.vn';  
  this.protocol = 'http://';  
  this.apiEndPoint = 'localhost:4100/api/';  
}
```

- Xây dựng nginx.conf file

+ Cài đặt thư mục gốc

+ Cài đặt file index

+ Điều hướng truy cập dựa vào đường dẫn

+ “kubedemo-api” là tên của service trong cụm được deploy bằng kubernetes, sẽ được trình bày khi xây dựng file deploy.yaml

```
nginx.conf X
front-end > nginx.conf
1  events{}
2
3  http {
4
5      include /etc/nginx/mime.types;
6
7      server {
8          listen 80;
9          server_name localhost;
10         root /usr/share/nginx/html;
11         index index.html;
12
13         location / {
14             try_files $uri $uri/ /index.html;
15         }
16
17         location /api{
18             proxy_pass http://kubedemo-api;
19         }
20     }
21 }
22
23
```

- Xây dựng Dockerfile:

- + Được đặt ở thư mục frone-end
- + Bao gồm cài đặt của nginx từ file nginx.conf
- + Bỏ tất cả nội dung front-end vào thư mục gốc của nginx

```
Dockerfile X
front-end > Dockerfile
1 FROM node:latest AS build
2 WORKDIR /Kubedemo
3 COPY package*.json /Kubedemo/
4 RUN npm install
5 COPY ./ /Kubedemo/
6 ARG configuration=prod
7
8 RUN npm run build:$configuration
9
10 FROM nginx:latest
11 COPY --from=build /Kubedemo/nginx.conf /etc/nginx/nginx.conf
12 COPY --from=build /Kubedemo/dist/kubedemo-app /usr/share/nginx/html
13
```

- Xây dựng docker image

+ Từ vị trí Dockerfile chạy lệnh:

**docker build -t kubefrontend:1.0 .** (có dấu chấm cuối)

+ Kết quả

```
[+] Building 159.1s (15/15) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 34B 0.0s
=> [internal] load metadata for docker.io/library/nginx:latest 2.8s
=> [internal] load metadata for docker.io/library/node:latest 3.8s
=> [stage-1 1/3] FROM docker.io/library/nginx:latest@sha256:4cf620a5c81390ee209398ecc18e5fb9dd0f5155cd82adcbae53 0.0s
=> [build 1/6] FROM docker.io/library/node:latest@sha256:be68c4603ebcf07b3910ebf66d290d4004091ad61fd8ea579a6689 16.4s
=> => resolve docker.io/library/node:latest@sha256:be68c4603ebcf07b3910ebf66d290d4004091ad61fd8ea579a6689a9244 0.0s
=> => sha256:7e5c5674d9325eee6fe859bf4da2cca67f02ca9e5e05d2f780be759325d73ea9 2.21kB / 2.21kB 0.0s
=> => sha256:8bc1fccfb49476ac42358e2a870c3997ad943175404d716e21838af436188796 7.81kB / 7.81kB 0.0s
=> => sha256:be68c4603ebcf07b3910ebf66d290d4004091ad61fd8ea579a6689a9244ff3 776B / 776B 0.0s
=> => sha256:d1c7d36631dcbda3a36b0eadfcbe04a97bc4f564e71f9903cc5b9bfd40e088e6 33.67MB / 33.67MB 13.7s
=> => sha256:65b46ea2c99d819df19073bfaffbfa539cd8e2c88799c73afa5c1ed35f1d7a41 2.38MB / 2.38MB 2.6s
=> => sha256:4e9083670c622ed68d8886506a47426010d0ea25725af53215ebd6361b9dbfa 294B / 294B 0.6s
=> => extracting sha256:d1c7d36631dcbda3a36b0eadfcbe04a97bc4f564e71f9903cc5b9bfd40e088e6 1.7s
=> => extracting sha256:65b46ea2c99d819df19073bfaffbfa539cd8e2c88799c73afa5c1ed35f1d7a41 0.1s
=> => extracting sha256:4e9083670c622ed68d8886506a47426010d0ea25725af53215ebd6361b9dbfa 0.0s
=> [internal] load build context 0.2s
=> => transferring context: 736.97kB 0.2s
=> [build 2/6] WORKDIR /Kubedemo 0.4s
=> [build 3/6] COPY package*.json /Kubedemo/ 0.1s
=> [build 4/6] RUN npm install 65.9s
=> [build 5/6] COPY ./ /Kubedemo/ 0.1s
=> [build 6/6] RUN npm run build:prod 70.7s
=> CACHED [stage-1 2/3] COPY --from=build /Kubedemo/nginx.conf /etc/nginx/nginx.conf 0.0s
=> CACHED [stage-1 3/3] COPY --from=build /Kubedemo/dist/kubedemo-app /usr/share/nginx/html 0.0s
=> exporting to image 0.1s
=> => exporting layers 0.0s
=> => writing image sha256:a33059b86a44920d98910ca43cb317e1189caf265b20586c1f70b5c3ceb63586 0.0s
=> => naming to docker.io/library/kubefrontend:1.0 0.0s
```

+ Docker image được tạo trên Docker Desktop

kubefrontend	1.0	a33059b86a44	about 1 hour ago	133.27 MB
--------------	-----	--------------	------------------	-----------

### 3.3.3. Deploy ứng dụng lên Kubernetes

- Xây dựng deploy.yaml file



- + Deployment backend: tên kubedemo-api, sử dụng image kubebackend:1.0, sử dụng cổng kết nối 80
- + Service backend: tên kubedemo-api, trỏ tới deployment backend với cổng kết nối 80
- + Deployment frontend: tên kubedemo-ui, sử dụng image kubefrontend:1.0, sử dụng cổng kết nối 80
- + Service frontend: tên kubedemo-ui, trỏ tới deployment frontend với cổng kết nối 80, loại là **LoadBalancer** với cổng kết nối công khai là 4100

```

! deploy.yaml X
! deploy.yaml > apiVersion
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: kubedemo-api
5  spec:
6    selector:
7      matchLabels:
8        app: kubedemo-api
9    template:
10     metadata:
11       labels:
12         app: kubedemo-api
13     spec:
14       containers:
15       - name: kubedemo-api
16         image: kubebackend:1.0
17         resources:
18           limits:
19             memory: "128Mi"
20             cpu: "500m"
21         ports:
22         - containerPort: 80
23           name: http
24 ---
25 apiVersion: v1
26 kind: Service
27 metadata:
28   name: kubedemo-api
29 spec:
30   selector:
31     app: kubedemo-api
32   ports:
33   - port: 80
34     targetPort: http
35 ---
36 ---
37 apiVersion: apps/v1
38 kind: Deployment
39 metadata:
40   name: kubedemo-ui
41 spec:
42   selector:
43     matchLabels:
44       app: kubedemo-ui
45   template:
46     metadata:
47       labels:
48         app: kubedemo-ui
49     spec:
50       containers:
51       - name: kubedemo-ui
52         image: kubefrontend:1.0
53         resources:
54           limits:
55             memory: "128Mi"
56             cpu: "500m"
57         ports:
58         - containerPort: 80
59 ---
60 apiVersion: v1
61 kind: Service
62 metadata:
63   name: kubedemo-ui
64 spec:
65   selector:
66     app: kubedemo-ui
67   type: LoadBalancer
68   ports:
69   - port: 4100
70     targetPort: 80
71
72

```

- Thực thi deploy ứng dụng:


+ Từ vị trí deploy.yaml file chạy lệnh:

```
kubectl apply -f deploy.yaml
```

+ Kết quả

```
deployment.apps/kubedemo-api created
service/kubedemo-api created
deployment.apps/kubedemo-ui created
service/kubedemo-ui created
```

+ Các container đã chạy trên Docker Desktop

	k8s_POD_kubedemo-api-59b8fb99cf-znlh9_default_7da27efb-76e3-41f4-b220-baa989f326bf_0	k8s.gcr.io/pause:3.2
RUNNING		
	k8s_POD_kubedemo-ui-6c564bb7b5-2dl6x_default_01043ae0-d9a7-4569-8c9f-51327a2ed7bf_0	k8s.gcr.io/pause:3.2
RUNNING		
	k8s_kubedemo-ui_kubedemo-ui-6c564bb7b5-2dl6x_default_01043ae0-d9a7-4569-8c9f-51327a2ed7bf_0	sha256:a33059b86a44920d98910ca43cb317e1189caf265b20586c1f70b5c3ceb63586
RUNNING		
	k8s_kubedemo-api_kubedemo-api-59b8fb99cf-znlh9_default_7da27efb-76e3-41f4-b220-baa989f326bf_0	sha256:1ae0cda6437985f60694cdde7911df26a26c601c3639830dd5a5973fba2b1b92
RUNNING		

+ Trang web có thể truy cập thông qua cổng kết nối 4100

KubedemoApp

localhost:4100

Random.tanphuc

Integer

List Randomizer

Decimal

Strings

String count:

Character by string count:

Options:

☐ Include digits

☐ Include upper case

☐ Include lower case

☐ Allow duplicate strings

Generate random number

Result:

Copyright © - HCMC University of Technology and Education - Cloud Computing - Kubedemo

## **CHƯƠNG 4: KẾT LUẬN**

### **4.1. Kết quả đạt được**

- Xây dựng được ứng dụng: trang web sinh ngẫu nhiên đơn giản
- Deploy được ứng dụng đã xây dựng lên Kubernetes

### **4.2. Các hạn chế**

Nhóm thực hiện đề nhằm tìm hiểu, nâng cao kiến thức không tránh khỏi những sai sót, hạn chế, bất cập. Một số hạn chế như:

- Ứng dụng còn tương đối đơn giản
- Đề tài chỉ thực hiện trên localhost

### **4.3. Hướng phát triển**

Áp dụng để thực hiện các ứng dụng thực tế và deploy lên các dịch vụ điện toán đám mây.

## TÀI LIỆU THAM KHẢO

- [1]. [Điện toán đám mây, Wikipedia](#)
- [2]. [Docker là gì? Tìm hiểu về Docker, topdev.vn](#)
- [3]. [Kubernetes là gì? Cùng tìm hiểu cách hoạt động, topdev.vn](#)