

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

~~~~~\*~~~~~



**ĐỒ ÁN MÔN HỌC**

**Đề tài: Deploy a microservice application on  
Kubernetes in an on-premise environment**

**Môn học: Hệ tính toán phân bố**

**Sinh viên thực hiện:** Tô Công Quân - 22521190  
Nguyễn Đặng Khánh Quốc - 22521212  
Nguyễn Đức Toàn - 22521490  
Trần Văn Thuận - 22521448

**Giảng viên hướng dẫn:** ThS. Bùi Thanh Bình

**TP.Hồ Chí Minh - 2025**

# LỜI CẢM ƠN

Lời đầu tiên, cho phép tập thể Nhóm 6 đến từ lớp NT533.P21 xin gửi lời cảm ơn và tri ân sâu sắc đến thầy Bùi Thanh Bình - giảng viên môn Hệ tính toán phân bố, vì sự tận tâm giúp đỡ, chỉ bảo và hướng dẫn trong quá trình thực hiện đề tài. Những hướng dẫn của thầy đóng vai trò quan trọng trong việc hoàn thiện đồ án này. Chúng em chân thành cảm ơn thầy vì kiến thức và kinh nghiệm mà thầy đã chia sẻ với chúng em.

Chúng em cũng muốn gửi lời cảm ơn đến tất cả các thành viên trong nhóm đồ án. Sự đóng góp và nỗ lực của mỗi người trong việc tìm kiếm tài liệu, đưa ra ý tưởng và hoàn thiện đề tài.

Cuối cùng, vì thời gian và năng lực có hạn nên không thể tránh khỏi sai sót trong khi thực hiện đồ án học tập của chúng em. Rất mong sự góp ý và bổ sung của thầy và các bạn để đề tài chúng em trở nên hoàn thiện hơn. Một lần nữa, chân thành cảm ơn thầy và tất cả mọi người đã theo dõi đề tài này.

# MỤC LỤC

|                                                |     |
|------------------------------------------------|-----|
| LỜI CẢM ƠN .....                               | II  |
| MỤC LỤC .....                                  | III |
| TÓM TẮT .....                                  | III |
| I. TỔNG QUAN .....                             | 1   |
| 1. Kubernetes .....                            | 1   |
| 1.1. Khái niệm .....                           | 1   |
| 1.2. Kiến trúc .....                           | 1   |
| 1.3. Các chứng năng chính .....                | 2   |
| 1.4. So sánh Kubernetes và Docker swarm .....  | 2   |
| 2. Microservices .....                         | 3   |
| 2.1. Khái niệm microservices .....             | 3   |
| 2.2. Kiến trúc .....                           | 4   |
| 2.3. So sánh Microservices và Monolithic ..... | 4   |
| II. MÔ HÌNH TRIỂN KHAI .....                   | 5   |
| III. TRIỂN KHAI .....                          | 6   |
| 1. Cài đặt cụm Kubernetes .....                | 6   |
| 2. Cài đặt công cụ quản lý K8s (Rancher) ..... | 9   |
| 3. Cài đặt Ingress .....                       | 12  |
| 4. Triển khai ứng dụng .....                   | 15  |
| IV. KẾT LUẬN .....                             | 16  |
| BẢNG PHÂN CÔNG .....                           | 17  |
| TÀI LIỆU THAM KHẢO .....                       | 17  |

# TÓM TẮT

Đề tài tập trung triển khai một ứng dụng microservice trên nền tảng Kubernetes trong môi trường on-premise. Trước hết, nhóm tìm hiểu tổng quan về Kubernetes và microservices, bao gồm kiến trúc, chức năng, cũng như so sánh với các giải pháp khác như Docker Swarm và Monolithic. Sau đó, nhóm tiến hành cài đặt cụm Kubernetes từ các bước chuẩn bị server, cấu hình mạng, cài đặt các thành phần cơ bản như containerd, kubeadm, kubectl. Công cụ quản lý Rancher được sử dụng để đơn giản hóa việc quản lý cụm. Nhóm cũng triển khai Ingress controller để định tuyến truy cập và sử dụng Nginx làm Load Balancer. Cuối cùng, ứng dụng microservices được đóng gói, đẩy lên Docker Hub và triển khai thành công trên hệ thống Kubernetes đã xây dựng.

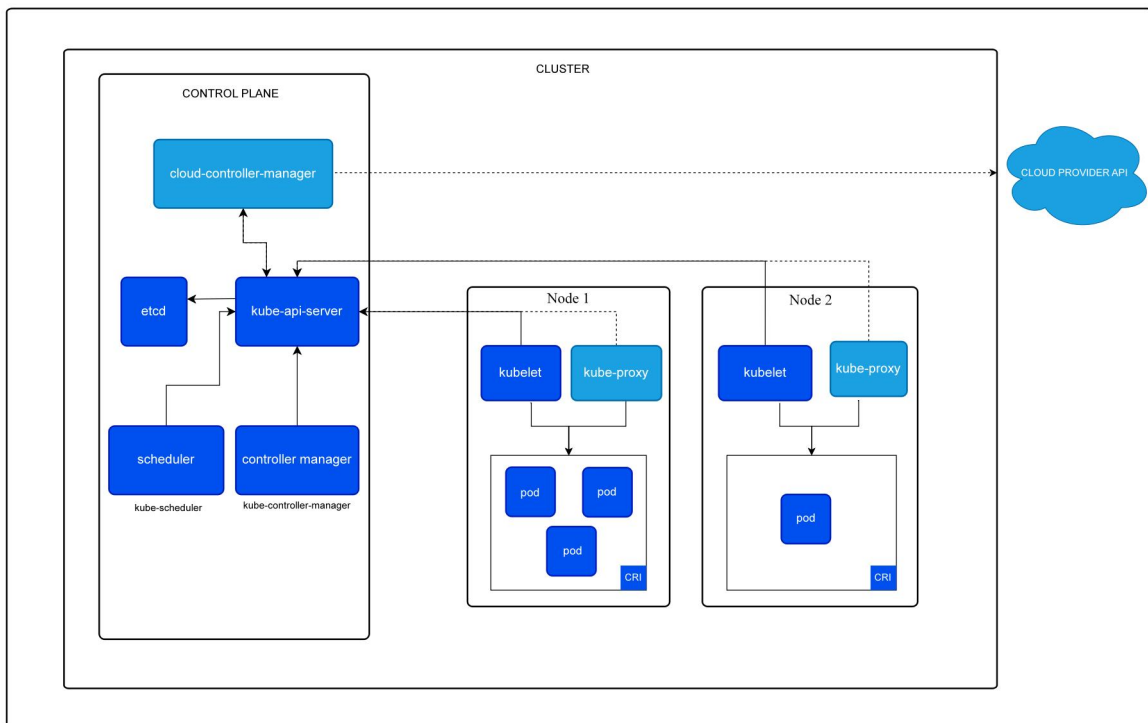
# I. TỔNG QUAN

## 1. Kubernetes

### 1.1. Khái niệm

Được biết đến với một cái tên khác là K8s, là một hệ thống mã nguồn mở cho việc triển khai, mở rộng và quản lý một cách tự động những ứng dụng dưới dạng container.

### 1.2. Kiến trúc



Hình 1: Kiến trúc Kubernetes

- Master node (control plane): được hiểu như là server điều khiển các node trong cụm kubernetes hoạt động. Gồm:
  - ◆ API server: Thành phần giúp cho các thành phần khác bên trong cụm K8s có thể giao tiếp được với nhau
  - ◆ Scheduler: Có chức năng lập lịch làm việc, triển khai cho các ứng dụng bên trong

- ◆ Controller manager: Quản lý hoạt động của tất cả worker trong K8s, bất kể còn hay đã mất
- ◆ Etcd: Là hệ cơ sở dữ liệu của K8s
- Node: hay còn được gọi là worker có chức năng thực hiện những tác vụ được yêu cầu, được giao.
  - ◆ Kuberlet: giúp cho node giao tiếp được với Master thông qua API server để hoạt động trơn tru hơn
  - ◆ Kube-proxy: Giúp cho các pod trong node có thể giao tiếp với bên ngoài
  - ◆ Pod: là đơn vị nhỏ nhất của một node, mỗi node chứa một hoặc nhiều container. Trong thực tế, mỗi pod thường đại diện cho một container, người ta hay gọi pod là server

### 1.3. Các chức năng chính

- Cân bằng tải: K8s quản lý nhiều Docker host bằng cách tạo các container cluster (cụm container). Ngoài ra, khi chạy một container trên Kubernetes, việc triển khai replicas (tạo các bản sao giống nhau) có thể đảm bảo cân bằng tải (load balancing) tự động và tăng khả năng chịu lỗi. Đồng thời nhờ có load balancing mà cũng có thể thực hiện autoscaling - tự động tăng giảm số lượng replicas.
- Tự phát hành và tự thu hồi: Docker host còn gọi là Node. Khi sắp xếp các container vào các Node này, có các dạng workload khác nhau như Disk, CPU,...
- Quản lý cấu hình: K8s có chức năng gọi là Service cung cấp load balancing cho một nhóm container cụ thể. Ngoài việc tự động thêm và xóa tại thời điểm scale, nó tự động ngắt kết nối trong trường hợp container bị lỗi.

### 1.4. So sánh Kubernetes và Docker swarm

|  | Kubernetes | Docker swarm |
|--|------------|--------------|
|--|------------|--------------|

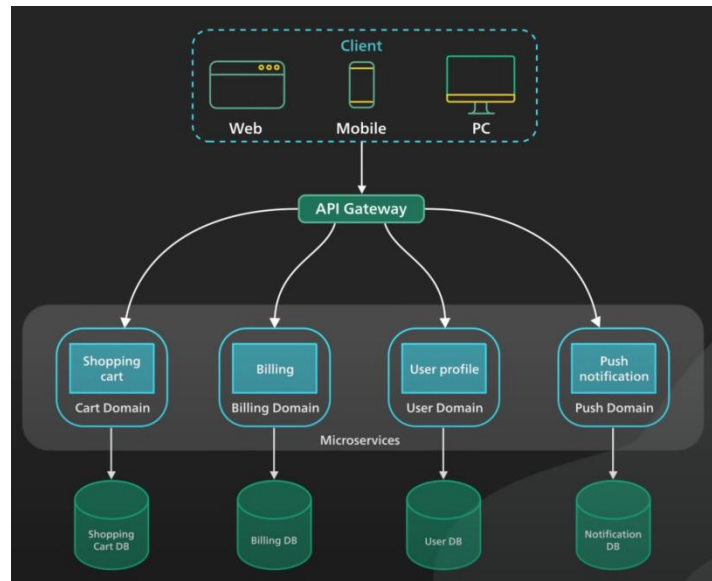
|                             |                                                                           |                                  |
|-----------------------------|---------------------------------------------------------------------------|----------------------------------|
| Cài đặt, cấu hình           | Khó cài đặt, cấu hình hơn                                                 | Cài đặt, cấu hình dễ hơn         |
| Mục đích                    | Quản lý cụm lớn, phức tạp                                                 | Quản lý cụm nhỏ                  |
| Cách quản lý                | Deployment, pod, service,...                                              | Sử dụng file service và task     |
| Tính sẵn sàng, tính mở rộng | Rất mạnh, tự động mở rộng và thay thế pod khi bị chết (khả năng chịu lỗi) | Không có autoscaling             |
| Giám sát                    | Có tích hợp giám sát và hỗ trợ tích hợp các công cụ giám sát bên thứ ba   | Cần ứng dụng bên thứ ba          |
| Load balancing              | Cần kết hợp service và Ingress                                            | Được hỗ trợ xây dựng trong swarm |

## 2. Microservices

### 2.1. Khái niệm microservices

Microservice là một kiểu kiến trúc phần mềm. Các module trong phần mềm này được chia thành các service rất nhỏ (microservice). Mỗi service sẽ được đặt trên một server riêng, vì thế dễ dàng để nâng cấp và mở rộng ứng dụng.

## 2.2. Kiến trúc



Hình 2: Kiến trúc microservices application

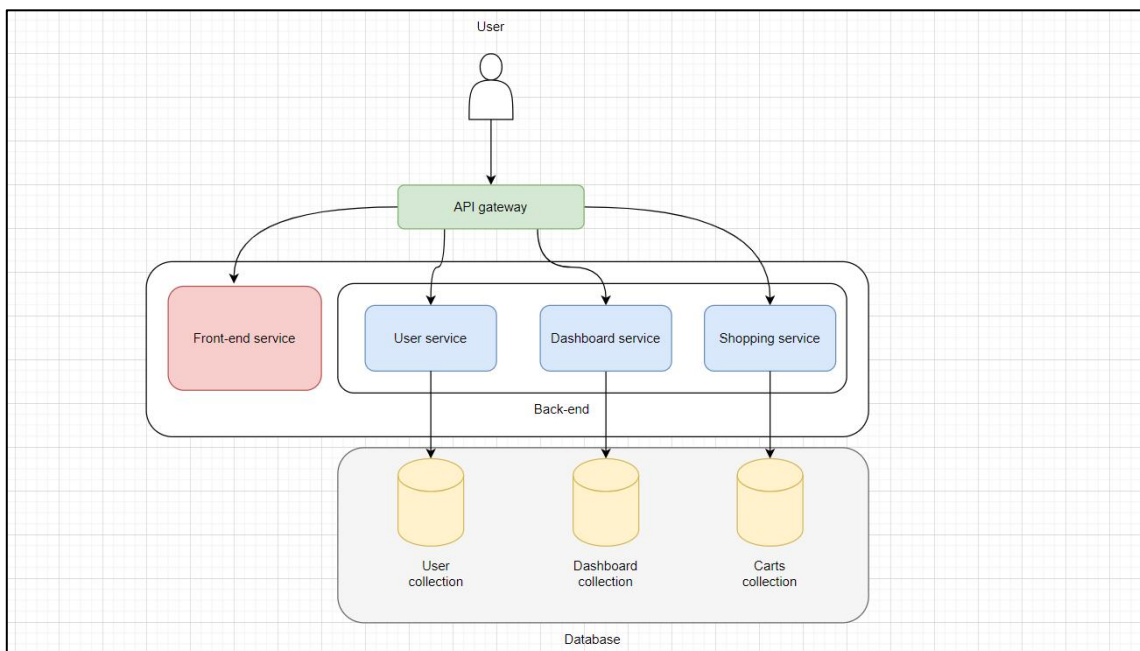
Mỗi module được chia thành các service riêng biệt, mỗi service kết nối với nơi lưu dữ liệu riêng của nó

## 2.3. So sánh Microservices và Monolithic

|              | Microservice                                      | Monolithic                                 |
|--------------|---------------------------------------------------|--------------------------------------------|
| Đặc điểm     | Chia nhỏ thành nhiều dịch vụ độc lập              | Là một khối duy nhất                       |
| Triển khai   | Triển khai từng service riêng biệt                | Triển khai toàn bộ ứng dụng một lần        |
| Độ phức tạp  | Phức tạp, cần thiết kế giao tiếp giữa các service | Ít phức tạp ban đầu, dễ phát triển lúc nhỏ |
| Tính ổn định | Lỗi service này không ảnh hưởng đến service kia   | Một lỗi có thể làm sập cả một hệ thống     |
| Tình bảo trì | Dễ bảo trì                                        | Khó bảo trì                                |



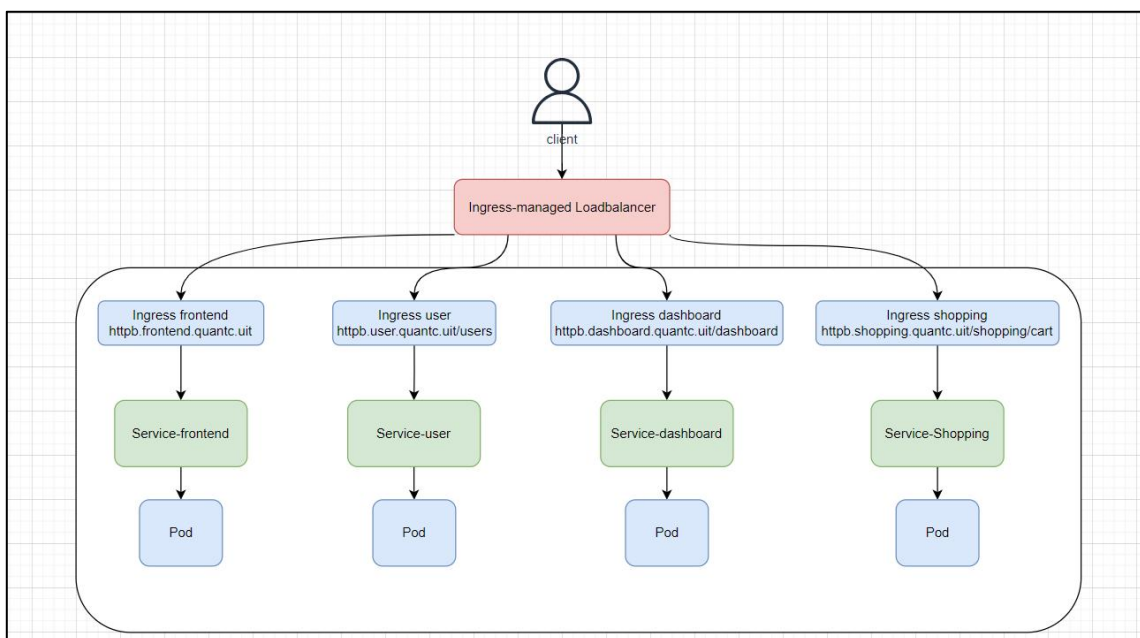
## II. MÔ HÌNH TRIỂN KHAI



Hình 3: Kiến trúc ứng dụng

Service frontend và các service backend đều được build thành images thông qua dockerfile, sau đó sẽ được đẩy lên Docker hub.

Cụ thể từng bước đóng gói ở trong file này: [link](#)



Hình 4: Mô hình triển khai

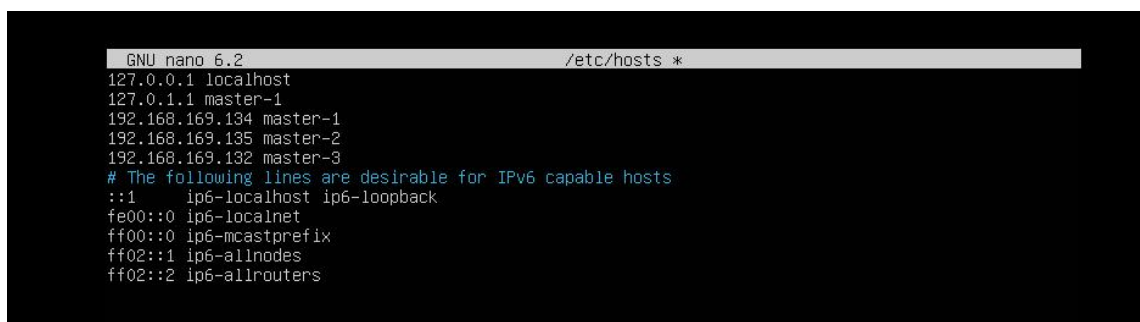
### III. TRIỂN KHAI

#### 1. Cài đặt cụm Kubernetes

| Server              | IP              | Host                  | Role          |
|---------------------|-----------------|-----------------------|---------------|
| Master-1            | 192.168.169.134 |                       | Control-plane |
| Master-2            | 192.168.169.135 |                       | worker        |
| Master-3            | 192.168.169.132 |                       | worker        |
| Rancher-server      | 192.168.169.136 | rancher.quantc.uit    |               |
| Loadbalancer-Server | 192.168.169.138 | <hostname>.quantc.uit |               |
| Database-Server     | 192.168.169.129 |                       |               |

Bảng thống số các server

- Ở master 1, thêm 3 ip và username của 3 VM vào file **/etc/hosts** theo dạng <ip> <tên user>

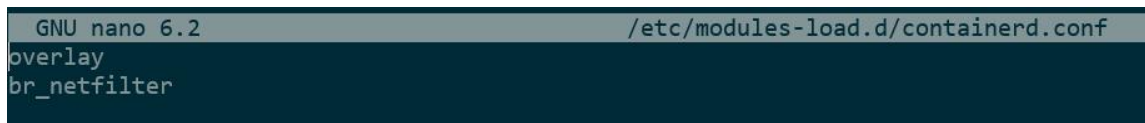


```
GNU nano 6.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 master-1
192.168.169.134 master-1
192.168.169.135 master-2
192.168.169.132 master-3
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Hình 5: Thêm các host

- Chạy lệnh apt update và apt upgrade cho cả 3 server
- Tạo user riêng trên cả 3 server bằng lệnh **adduser**
- Thêm các user đã tạo vào group sudo để có đủ quyền thực thi (lệnh: **usermod -aG sudo <tên user>**), sau đó chuyển sang user đó bằng lệnh **su - <tên user>**

- Tắt tạm thời swap các node (VM) bằng lệnh `sudo swapoff -a`. Nếu muốn tắt vĩnh viễn thì vào file `/etc/fstab` command lại dòng swap.img bằng lệnh `sudo sed -i '/swap.img/s/^/#/' /etc/fstab`
- Cài đặt tài nguyên cho cụm bằng cách cấu hình module kernel, chạy lệnh nano `/etc/modules-load.d/containerd.conf` thêm 2 dòng:



```
GNU nano 6.2 /etc/modules-load.d/containerd.conf
overlay
br_netfilter
```

Hình 6: Cấu hình module kernel

- Sau đó tải module kernel bằng hai lệnh: `sudo modprobe overlay` và `sudo modprobe br_netfilter`

- Cấu hình hệ thống mạng bằng 3 câu lệnh:

```
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https
ca-certificates
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/trusted.gpg.d/docker.gpg
```

```
Sudo          add-apt-repository          "deb          [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

- Áp dụng cấu hình và kiểm tra bằng lệnh `sudo systemctl --system`

- Cài đặt các gói cần thiết và thêm kho Docker

```
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https
ca-certificates
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/trusted.gpg.d/docker.gpg
```

```
sudo          add-apt-repository          "deb          [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

- Cập nhật lại hệ thống và cài đặt containerd bằng lệnh: `sudo apt update -y` và `sudo apt install -y containerd.io`

- Cấu hình cho container:

```
containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
```

```
sudo sed -i 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml
```

- Khởi động lại và enable container: `sudo systemctl restart containerd` và `sudo systemctl enable containerd`

- Thêm kho lưu trữ cho Kubernetes :

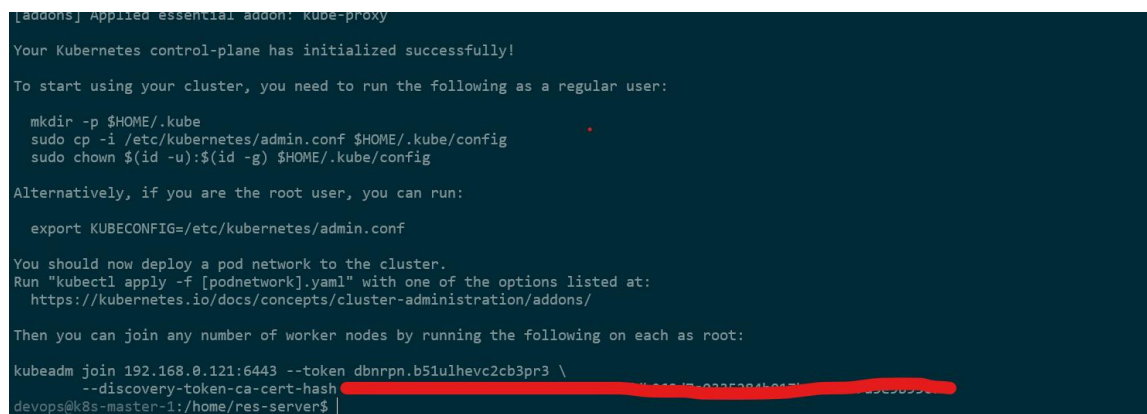
```
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

- Cài đặt các gói Kubernetes, sử dụng `apt-mark hold` để giữ lại version để tránh bị lỗi không tương thích bằng các câu lệnh: `sudo apt update -y`, `sudo apt install -y kubelet kubeadm kubectl` và `sudo apt-mark hold kubelet kubeadm kubectl`

- Sử dụng lệnh `sudo kubeadm init phase kubelet-start` để tự tạo các file .yaml nếu bị thiếu trên server master-1

- Trên server làm master khởi tạo kubeadm bằng lệnh `sudo kubeadm init`



```
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.0.121:6443 --token dbnrpn.b51ulhevc2cb3pr3 \
--discovery-token-ca-cert-hash sha256:833599464281...
devops@k8s-master-1:/home/res-server$
```

Hình 7: Khởi tạo kubeadm

- Khi chạy lệnh thành công sẽ có các câu lệnh hướng dẫn để thực hiện và token. Thực hiện 3 câu lệnh trên các máy. Để thêm 2 worker node vào cluster thì chạy lệnh kubeadm join như trên hình, sử dụng với sudo.

- Sau đó sử dụng lệnh kubectl get node trên master để xem trạng thái của cluster

```
devops@k8s-master-1:/home/res-server$ kubectl get nodes
NAME             STATUS    ROLES    AGE   VERSION
k8s-master-1     NotReady  control-plane  35m   v1.30.10
k8s-master-2     NotReady  <none>      44s   v1.30.10
k8s-master-3     NotReady  <none>      8s    v1.30.10
```

Hình 8: Kiểm tra trạng thái các node

- Ta có thể thấy status là NotReady vì cụm chưa có network, vì thế cần thêm network cho cụm bằng lệnh

kubectl apply -f  
<https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml>

- Kiểm tra lại các node

```
devops@k8s-master-1:/home/res-server$ kubectl get nodes
NAME             STATUS    ROLES    AGE   VERSION
k8s-master-1     Ready     control-plane  42m   v1.30.10
k8s-master-2     Ready     <none>      7m5s   v1.30.10
k8s-master-3     Ready     <none>      6m29s   v1.30.10
```

Hình 8: Kiểm tra trạng thái các node sau khi thêm network

## 2. Cài đặt công cụ quản lý K8s (Rancher)

- Đầu tiên, tạo một máy ảo server Rancher để lưu trữ dữ liệu cho cụm, tạo thêm một hard disk mới và thiết lập hostname cho server

- Format ổ đĩa mới vừa tạo nhưng chưa được apply bằng lệnh: sudo mkfs.ext4 -m 0 /dev/sdb

- Tạo một thư mục mới muốn mount ổ đĩa đó và Cấu hình để ổ cứng mới (thứ 2) map vào trong mount point - thư mục vừa tạo (ở đây là thư mục /data): echo "/dev/sdb /data ext4 defaults 0 0" | sudo tee -a /etc/fstab

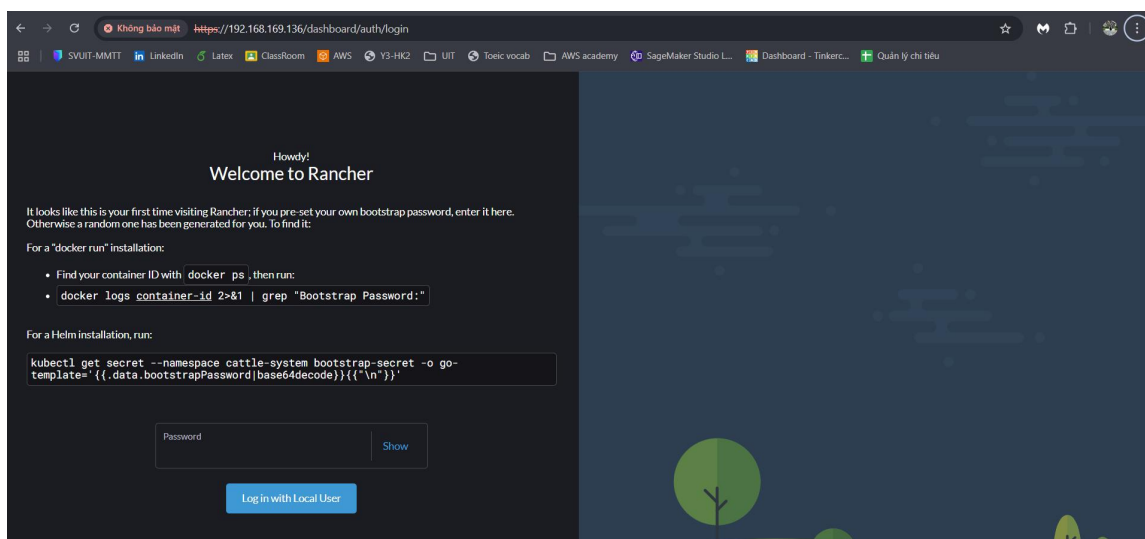
- Chạy lệnh mount -a, Update hệ thống và cài đặt docker

- Tạo một thư mục riêng cho rancher trong thư mục /data, tạo một docker-compose.yaml. Sau đó chạy lệnh **docker-compose up -d**

```
version: '3'
services:
  rancher-server:
    image: rancher/rancher:v2.9.2
    container_name: rancher-server
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /data/rancher/data:/var/lib/rancher
    privileged: true
```

Hình 9: Tạo docker-cpmpose.yaml cho rancher

- Sau khi chạy thành công, truy cập trang web với ip của server hoặc hostname

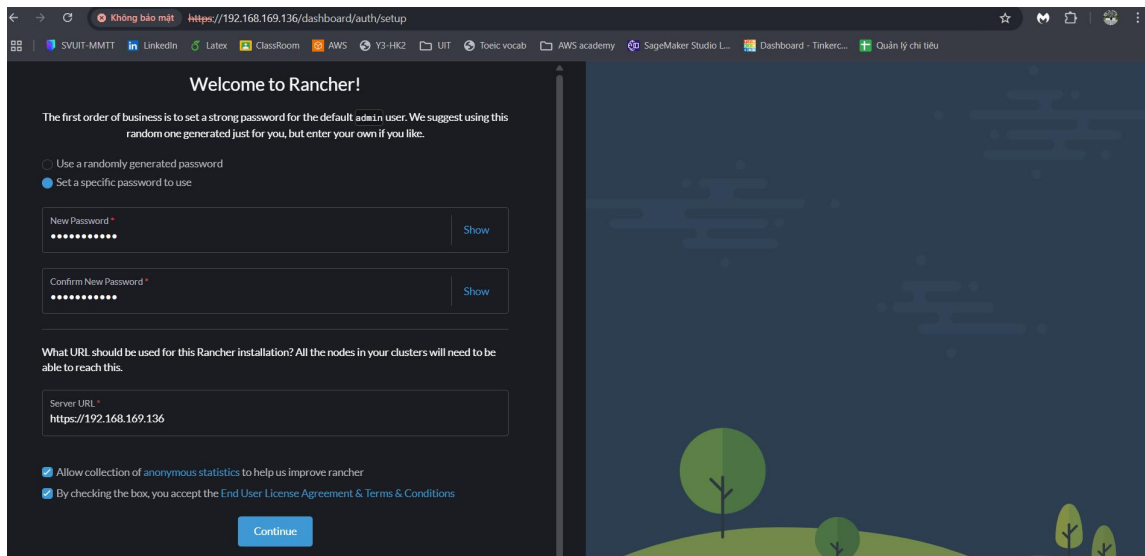


Hình 10: Truy cập Rancher server

- Dùng lệnh hướng dẫn để lấy password

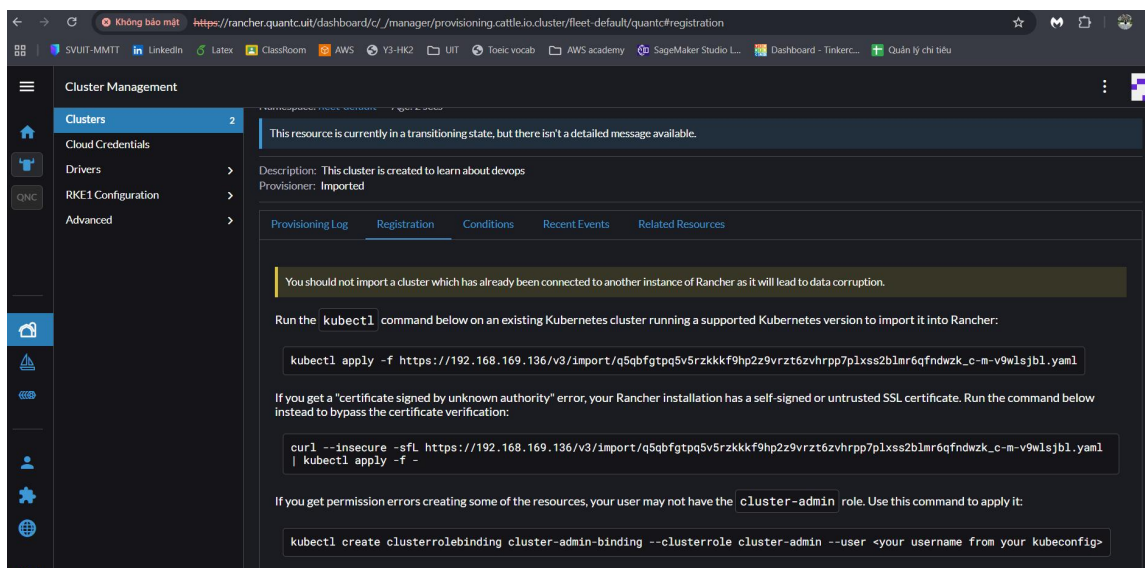
```
root@rancher:/data/rancher# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
4688fd50f983   rancher/rancher:v2.9.2   "entrypoint.sh"         3 minutes ago   Up 3 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp
rancher-server
root@rancher:/data/rancher# docker logs rancher-server 2>&1 | grep "Bootstrap Password:"
2025/03/21 13:25:22 [INFO] Bootstrap Password: 88sw9gmq8xxgsffffnbs6shf9t4ntp1zhm16f4k5rp7zk5rv2kwzj7w
root@rancher:/data/rancher#
```

Hình 11: Chạy lệnh lấy password



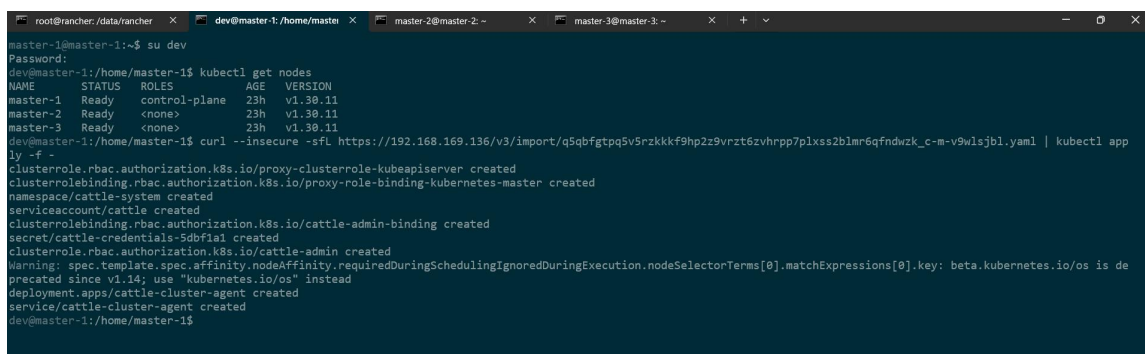
Hình 12: Đăng nhập Rancher

- Import k8s lên rancher quản lý, vì đây là on-premise nên chọn Generic. Đặt tên đúng với mục đích và chọn Create



Hình 13: Import cụm K8s

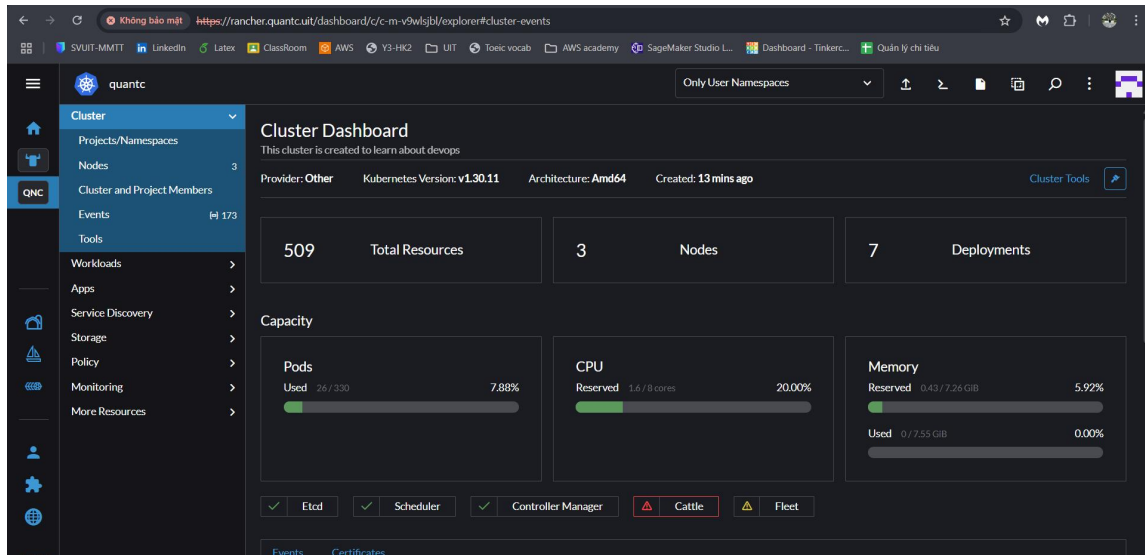
- Vì cert tự ký nên sẽ chạy lệnh dòng thứ 2 trên master





Hình 14: Chạy lệnh để import

- Chờ cluster trên Rancher chuyển sang trạng thái active ta có thể xem thông tin về cụm k8s



Hình 15: Import K8s thành công

### 3. Cài đặt Ingress

- Ingress phải đi kèm với Ingress controller, nếu không sẽ vô nghĩa, trong đó nhóm chọn ingress nginx

- Trên node master-1 cài đặt helm bằng các lệnh

`wget https://get.helm.sh/helm-v3.16.2-linux-amd64.tar.gz`

`tar xvf helm-v3.16.2-linux-amd64.tar.gz`

`sudo mv linux-amd64/helm /usr/bin/`

- Thêm repo ingress nginx, update repo, sau đó pull ingress-nginx và giải nén file .tgz



```

master-1@master-1:~$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
"ingress-nginx" has been added to your repositories
master-1@master-1:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
Update Complete. Happy Helming!
master-1@master-1:~$ helm repo list
NAME      URL
ingress-nginx https://kubernetes.github.io/ingress-nginx
master-1@master-1:~$ helm search repo nginx
NAME                CHART VERSION    APP VERSION    DESCRIPTION
ingress-nginx/ingress-nginx  4.12.1          1.12.1        Ingress controller for Kubernetes using NGINX a...
master-1@master-1:~$ helm pull ingress-nginx/ingress-nginx
master-1@master-1:~$ ls
helm-v3.16.2-linux-amd64.tar.gz  ingress-nginx-4.12.1.tgz  linux-amd64
master-1@master-1:~$ tar -xzf ingress-nginx-4.12.1.tgz
master-1@master-1:~$ ls
helm-v3.16.2-linux-amd64.tar.gz  ingress-nginx  ingress-nginx-4.12.1.tgz  linux-amd64
master-1@master-1:~$ cd ingress-nginx/
master-1@master-1:~/ingress-nginx$ ls
changelog  Chart.yaml  ci  OWNERS  README.md  README.md.gotmpl  templates  tests  values.yaml
master-1@master-1:~/ingress-nginx$ nano values.yaml

```

Hình 16: Pull ingress-nginx

- Chỉnh sửa file values.yaml, sửa type từ LoadBalancer → NodePort

```

configMapKey: ""
service:
  # -- Enable controller services or not. This does not influence the creation of either the admission webhook or the metrics service
  enabled: true
  external:
    # -- Enable the external controller service or not. Useful for internal-only deployments.
    enabled: true
    # -- Annotations to be added to the external controller service. See `controller.service.internal.annotations` for annotations to
    annotations: {}
    # -- Labels to be added to both controller services.
    labels: {}
    # -- Type of the external controller service.
    # Ref: https://kubernetes.io/docs/concepts/services-networking/service/#publishing-services-service-types
    type: NodePort
    # -- Pre-defined cluster internal IP address of the external controller service. Take care of collisions with existing services.
    # This value is immutable. Set once, it can not be changed without deleting and re-creating the service.
    # Ref: https://kubernetes.io/docs/concepts/services-networking/service/#choosing-your-own-ip-address
    clusterIP: ""

```

Hình 17: Chỉnh sửa type

- Gán expose port cụ thể tránh random port

```

# -- Declare the app protocol of the external HTTP and HTTPS listeners or not. Supersedes provider-specific annotations for declaring the backend protocol
# Ref: https://kubernetes.io/docs/concepts/services-networking/service/#application-protocol
appProtocol: true
nodePorts:
  # -- Node port allocated for the external HTTP listener. If left empty, the service controller allocates one from the configured node port range.
  http: "30080"
  # -- Node port allocated for the external HTTPS listener. If left empty, the service controller allocates one from the configured node port range.
  https: "30443"
  # -- Node port mapping for external TCP listeners. If left empty, the service controller allocates them from the configured node port range.
  # Example:
  # tcp:

```

Hình 18: Gán expose port

- Copy folder ingress-nginx vào trong /home/dev/
- Tạo một ns riêng cho ingress

```
dev@master-1:~$ helm -n ingress-nginx install ingress-nginx -f ingress-nginx/values.yaml ingress-nginx
NAME: ingress-nginx
LAST DEPLOYED: Tue Mar 25 14:15:52 2025
NAMESPACE: ingress-nginx
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The ingress-nginx controller has been installed.
Get the application URL by running these commands:
  export HTTP_NODE_PORT=30080
  export HTTPS_NODE_PORT=30443
  export NODE_IP=$(kubectl get nodes --output jsonpath="{.items[0].status.addresses[1].address}")
  echo "Visit http://${NODE_IP}:${HTTP_NODE_PORT} to access your application via HTTP."
  echo "Visit https://${NODE_IP}:${HTTPS_NODE_PORT} to access your application via HTTPS."

An example Ingress that makes use of the controller:
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example
  namespace: foo
spec:
```

Hình 19: Cài đặt ingress nginx

- Tạo một server riêng cho LoadBalancer, vì sử dụng nginx là LoadBalancer nên cần tải nginx bằng lệnh: **apt install nginx**
- Đổi cấu hình mặc định của nginx ở file default **/etc/nginx/sites-available/default**.  
Đổi Listen port từ 80 sang một port khác, ví dụ như 3131

```
# Default server configuration
#
server {
    listen 3131 default_server;
    listen [::]:3131 default_server;

    # SSL configuration
    #
```

Hình 20: Đổi listen port

- Tạo một file cấu hình nginx load balancer trong thư mục **/etc/nginx/conf.d/quantic.conf**. Với quantic.conf là file tự tạo, đặt tên sao cho có **.conf**

```

upstream my_servers {
    server 192.168.169.134:30080;
    server 192.168.169.135:30080;
    server 192.168.169.132:30080;
}

server {
    listen 80;

    location / {
        proxy_pass http://my_servers;
        proxy_redirect off;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

```

Hình 21: Cấu hình nginx load balancer

- Sau đó kiểm tra bằng lệnh `nginx -t`. Nếu kết quả trả về là ok và successful thì thành công. Sau đó restart lại nginx để apply cấu hình

```

root@loadbalancer-k8s:/home/lbl# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@loadbalancer-k8s:/home/lbl# |

```

Hình 22: Cấu hình thành công

## 4. Triển khai ứng dụng

Các file cấu hình và cách chạy câu lệnh để áp dụng cấu hình được mô tả chi tiết trong [file](#) này.

Link video demo:

Demo về thử nghiệm triển khai ứng dụng microservice: [link](#)

Demo về thử nghiệm xóa một service, xem các service khác có hoạt động bình thường: [link](#)

Demo về cách dùng Configmap và Secret trong dự án: [link](#)

Demo về auto scaling một Deployment: [link](#)

## IV. KẾT LUẬN

Qua quá trình thực hiện đề tài, nhóm đã có cơ hội tiếp cận thực tế với việc triển khai và vận hành hệ thống Kubernetes trong môi trường on-premise. Từ việc xây dựng cụm K8s, cài đặt công cụ quản lý Rancher, cấu hình Ingress, cho đến việc triển khai ứng dụng microservice, mỗi bước đều giúp nhóm hiểu sâu hơn về cách hoạt động của các thành phần. Mặc dù gặp phải một số khó khăn như lỗi cấu hình hay các vấn đề tương thích giữa các phiên bản phần mềm, nhưng thông qua việc tìm hiểu và khắc phục, nhóm đã rèn luyện được kỹ năng giải quyết vấn đề và tư duy hệ thống. Đây là nền tảng tốt để nhóm có thể ứng dụng vào các dự án thực tế trong tương lai. Nhóm cũng nhận thấy tầm quan trọng của việc chuẩn hóa quy trình triển khai và tối ưu tài nguyên trong hệ thống phân tán.

## BẢNG PHÂN CÔNG

| STT | Họ và tên              | MSSV     | Phân công                                                                        |
|-----|------------------------|----------|----------------------------------------------------------------------------------|
| 1   | Tô Công Quân           | 22521190 | Viết báo cáo, làm slide, cài đặt máy ảo, cài đặt kubernetes, triển khai ứng dụng |
| 2   | Nguyễn Đăng Khánh Quốc | 22521212 | Viết frontend service và đóng gói                                                |
| 3   | Nguyễn Đức Toàn        | 22521490 | Viết frontend service và đóng gói                                                |
| 4   | Trần Văn Thuận         | 22521448 | Viết backend service và đóng gói push lên Dockerhub                              |

## TÀI LIỆU THAM KHẢO

**1.Khoá học kubernetes từ cơ bản đến thực tế**

**2.Kubernetes Documents**

File write up về khoá học, có cả cài đặt hệ thống giám sát trên cụm K8s: [link](#)