**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN HỌC PHẦN**

Tên học phần: **ĐIỆN TOÁN ĐÁM MÂY (Cloud Computing), tiết 14 - 15 tối thứ 2, tiết 11 - 12 chiều thứ 7**

Tên đề tài Nhóm: **TÌM HIỂU VỀ KUBERNETES VÀ VIẾT ỨNG DỤNG DEMO**

**Giảng viên giảng dạy: Thầy Huỳnh Xuân Phụng \_ Giảng viên chính**

DANH SÁCH THÀNH VIÊN NHÓM

1. Trần Tiến Đức - 18133007
2. Huỳnh Thiên Long - 18133025

**Ngày nộp: ……./……./2021**

**TP.HCM, ngày …. Tháng …. năm 2021**

**Mục lục**

[I. Kubernetes (K8s) là gì? 4](#_Toc31606)

[II. Các hệ thống cung cấp Kubernetes cài đặt sẵn: 4](#_Toc2796)

[III. Các cách để cài đặt K8s 4](#_Toc19855)

[IV. Các đối thủ cạnh tranh 5](#_Toc16392)

[V. Kiến trúc 5](#_Toc20774)

[1. Master server 6](#_Toc23840)

[a) Etcd 6](#_Toc20583)

[b) API Server 6](#_Toc29026)

[c) Controller Manager Service 6](#_Toc7343)

[d) Scheduler Service 7](#_Toc13303)

[e) Dashboard (Không bắt buộc) 7](#_Toc5825)

[2. Node Server 7](#_Toc22510)

[a) Pod 7](#_Toc28026)

[b) Service (svc) 7](#_Toc8332)

[c) Persistent Volumes (PV) 8](#_Toc28514)

[d) Namespaces (Không gian tên) 9](#_Toc16686)

[e) Ingress rules 10](#_Toc28045)

[f) Network policies 11](#_Toc11547)

[g) Network 11](#_Toc25247)

[h) ConfigMaps and Secrets 11](#_Toc24310)

[i) Controllers 11](#_Toc26367)

[j) Helm - Trình quản lý gói của K8s 12](#_Toc7342)

[k) Dashboard 12](#_Toc880)

[l) Monitoring 12](#_Toc9258)

[m) Create 12](#_Toc15790)

[VI. Mô hình hệ thống được sử dụng: 12](#_Toc19961)

[VII. Tài liệu hướng dẫn sử dụng: 13](#_Toc22213)

[VIII. Vấn đề còn tồn tại: 15](#_Toc24265)

[IX. Hướng phát triển: 16](#_Toc12596)

[NGUỒN THAM KHẢO 17](#_Toc17794)

1. **Kubernetes (K8s) là gì?**

Kubernetes là một nền tảng nguồn mở, khả chuyển, có thể mở rộng để quản lý các ứng dụng được đóng gói và các service, giúp thuận lợi trong việc cấu hình và tự động hoá việc triển khai ứng dụng. Kubernetes là một hệ sinh thái lớn và phát triển nhanh chóng. Các dịch vụ, sự hỗ trợ và công cụ có sẵn rộng rãi.

Tên gọi Kubernetes có nguồn gốc từ tiếng Hy Lạp, có ý nghĩa là người lái tàu hoặc hoa tiêu. Google mở mã nguồn Kubernetes từ năm 2014. Kubernetes xây dựng dựa trên [một thập kỷ rưỡi kinh nghiệm mà Google có được với việc vận hành một khối lượng lớn workload trong thực tế](https://research.google/pubs/pub43438), kết hợp với các ý tưởng và thực tiễn tốt nhất từ cộng đồng.

# **Các hệ thống cung cấp Kubernetes cài đặt sẵn:**

* Google container engine (GKE): [https://cloud.google.com/container-engine/](https://cloud.google.com/container-engine/" \t "https://viblo.asia/p/_blank) (free 300$ cho tài khoản mới)
* CoreOS techtonic: [https://coreos.com/tectonic/](https://coreos.com/tectonic/" \t "https://viblo.asia/p/_blank)
* RedHat Openshift: [https://www.openshift.com/](https://www.openshift.com/" \t "https://viblo.asia/p/_blank)
* Amazon EKS (Elastic Kubernetes Service)

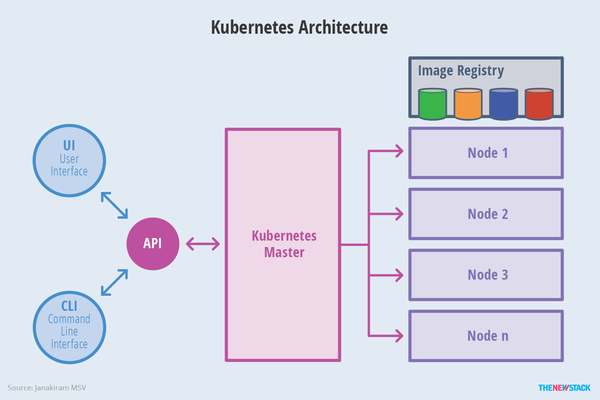
# **Các cách để cài đặt K8s**

K8s là một hệ thống, gồm nhiều thành phần tương tác với nhau. Tuy không tới nỗi phức tạp như cài một hệ thống cloud IaaS (Infranstructure as A Service - Kiến trúc hạ tầng như một dịch vụ) như OpenStack. Vậy là người ta có hàng chục cách khác nhau để cài K8s, đáng kể nhất có:

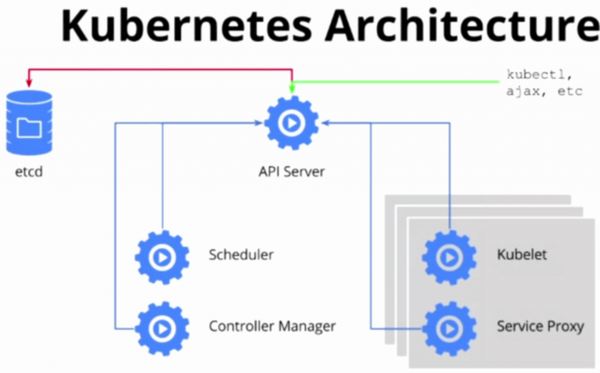
1. Minikube (Tích hợp sẵn trên Docker Desktop của Windows và MacOS chủ yếu triển khai cụm 1 node)
2. Kubeadm đang trong giai đoạn phát triển, để cài trên hệ thống máy vật lý / máy ảo dùng Ubuntu 16.04 hay CentOS 7
3. Kargo là phần mềm dựa trên Ansible (Phần mềm tự động hoá cấu hình) để cài trên rất nhiều nơi bao gồm cả máy vật lý/máy ảo/AWS/GCE
4. Dùng SaltStack để cài [https://github.com/kubernetes/kubernetes/tree/master/cluster/saltbase/](https://github.com/kubernetes/kubernetes/tree/master/cluster/saltbase/" \t "https://viblo.asia/p/_blank)
5. Cài bằng tay trên [CoreOS Container Linux](https://coreos.com/kubernetes/docs/latest/getting-started.html" \t "https://viblo.asia/p/_blank)
6. Kops để cài trên AWS (Amazon cloud)
7. **Các đối thủ cạnh tranh**

* Docker Swarm [https://docs.docker.com/engine/swarm/](https://docs.docker.com/engine/swarm/" \t "https://viblo.asia/p/_blank)
* Apache Mesos [https://mesos.apache.org/](https://mesos.apache.org/" \t "https://viblo.asia/p/_blank)

# **Kiến trúc**

K8s cluster bao gồm nhiều node, trên mỗi node sẽ cần chạy một "kubelet", đây là chương trình để chạy k8s. Cần một máy để làm "chủ" cluster, trên đó sẽ cài API server, scheduler ... Các máy còn lại sẽ chạy kubelet để sinh ra các container. 

## Master server



### Etcd

Có thể liên kết cài đặt với từng node thông qua etcd.

### API Server

Đúng theo tên gọi, đây chính là server cung cấp [Kubernetes API](https://github.com/GoogleCloudPlatform/kubernetes/blob/master/api/kubernetes.html" \t "https://viblo.asia/p/_blank). Nó có nhiệm vụ đặt Pod vào Node, đồng bộ hoá thông tin của Pod bằng REST API tiếp nhận cài đặt của pod/service/replicationController.

### Controller Manager Service

Được hiểu giống như “kube-controller manager”, nó quản lý tất cả các bộ điều khiển xử lý các tác vụ thông thường trong cluster. Chúng bao gồm Node Controller, Replication Controller, Endpoints Controller, and Service Account and Token Controllers. Chi tiết của các hoạt động này được ghi vào etcd, nơi controller manager theo dõi sự thay đổi thông qua API Server.

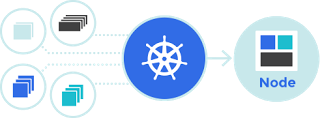
### Scheduler Service

Scheduler Service có trách nhiệm giám sát việc sử dụng tài nguyên trên mỗi máy chủ để đảm bảo rằng hệ thống không bị quá tải. Scheduler Service phải biết tổng số tài nguyên có sẵn trên mỗi máy chủ, cũng như các tài nguyên được phân bổ cho các khối lượng công việc hiện có được gán trên mỗi máy chủ.

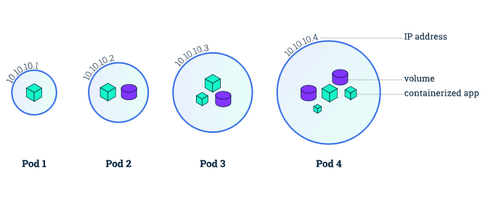
### Dashboard (Không bắt buộc)

Giao diện web Kubernetes giúp đơn giản hóa các tương tác của người dùng Kubernetes cluster với máy chủ API.

## Node Server

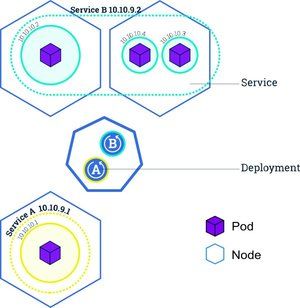


### Pod

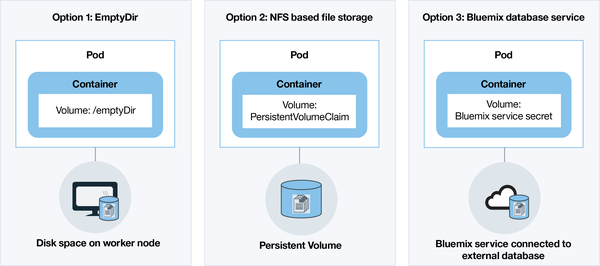
Pod là 1 nhóm (1 trở lên) các container thực hiện một mục đích nào đó, như là chạy software nào đó. Nhóm này chia sẻ không gian lưu trữ, địa chỉ IP với nhau. Pod thì được tạo ra hoặc xóa tùy thuộc vào yêu cầu của dự án. 

### Service (svc)

Vì các Pod có tuổi thọ ngắn, do vậy nó không đảm bảo về địa chỉ IP luôn cố định. Điều này khiến cho việc giao tiếp giữa các microservice trở nên khó khăn. Do đó, K8s giới thiệu về một dịch vụ, nó là một lớp nằm trên một số nhóm Pod. Nó được được gắn địa chỉ IP tĩnh và có thể trỏ domain vào dịch vụ này. Tại đây chúng ta có thể thực hiện cân bằng tải. Mỗi service sẽ được gán 1 domain do người dùng lựa chọn, khi ứng dụng cần kết nối đến service, ta chỉ cần dùng domain là xong. Domain được quản lý bởi hệ thống name server SkyDNS nội bộ của k8s - một thành phần sẽ được cài khi ta cài k8s.

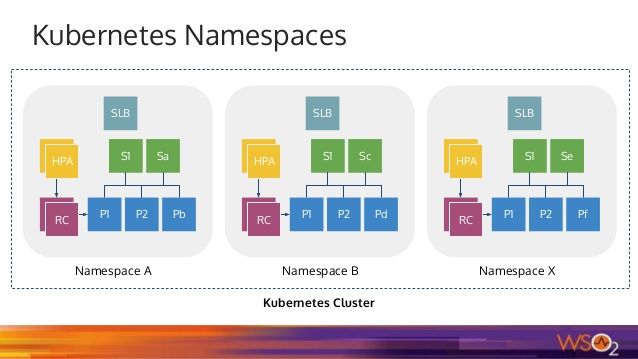


### Persistent Volumes (PV)

Bất kỳ ai làm container cũng cần hiểu rằng, ta không lưu dữ liệu trên container mà phải lưu nó vào một chỗ nào đó. Bởi khi container gặp sự cố hay khởi động lại thì dữ liệu cũng sẽ mất theo nó. Đây là chuyện dù dùng docker trực tiếp hay giải pháp khác K8s thì ta vẫn phải tính. Việc lưu dữ liệu của app trên container trực tiếp trên máy host là một giải pháp nhỏ lẻ. Vì nếu ta cho dữ liệu của pod của app A vào /var/lib/app/A, mà có 2 pod cho app A cùng được chạy trên máy đó thì chuyện gì xảy ra? Giải pháp k8s sử dụng là các hệ thống lưu trữ "network", tức lưu vào một hệ thống storage khác. Như NFS, GlusterFS, Ceph ... PV, là khái niệm để đưa ra một dung lượng lưu trữ THỰC TẾ 1GB, 10GB ... PVC là khái niệm ảo, đưa ra một dung lượng CẦN THIẾT, mà ứng dụng yêu cầu. Khi 1 PV thoả mãn yêu cầu của 1 PVC thì chúng "match" nhau, rồi "bound" (buộc / kết nối) lại với nhạu. Nếu tự cài K8s, hãy chuẩn bị sẵn giải pháp lưu trữ của bản thân. Nếu dùng sẵn Google cloud hay AWS, sẵn sàng để trả tiền. Kubernetes hỗ trợ nhiều kiểu volumes, như là: NFS, Ceph, GlusterFS, local directory, ... 

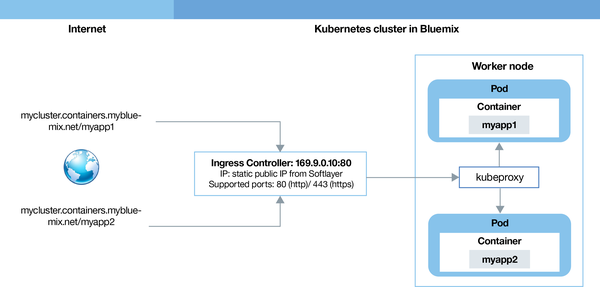
### Namespaces (Không gian tên)

Đây là một công cụ dùng để nhóm hoặc tách các nhóm đối tượng. Namespaces được sử dụng để kiểm soát truy cập, kiểm soát truy cập network, quản lý resource và quoting.



Nếu ta đặt service này là "web" lúc chạy production, còn lúc phát triển thì ta chạy nó ở đâu? Ta sẽ phải thay đỏi tên service. Namespace giải quyết vấn đề này. Mặc định các dịch vụ sẽ sử đụng namespace "default", nhưng ta có thể tạo namespace tuỳ ý. K8s sử dụng 1 namespace riêng : kube-system.

### Ingress rules

Dùng để quản lý network ra và vào các service và pod 

### Network policies

Định nghĩa các quy tắc truy cập mạng giữa các Pod bên trong Cluster.

### Network

Có nhiều loại phần mềm để triển khai container network, như Flannel, Weaver ... nếu ta dùng Google Cloud, vấn đề này không cần quan tâm.

### ConfigMaps and Secrets

Một phần mềm ít khi được khởi động và chạy luôn mà không cần cấu hình. ConfigMap là giải pháp để nhét 1 file config / đặt các environment var (Biến môi trường) hay thiết lập các tham số khi gọi câu lệnh. ConfigMap là một cục cấu hình, mà pod nào cần, thì chỉ định là nó cần - giúp dễ dàng chia sẻ file cấu hình. Ít ai muốn đặt mật khẩu vào file cấu hình, và chỉ có lập trình viên "tồi" mới hardcode mật khẩu vào code. Vậy nên K8s có "secret", để lưu trữ các mật khẩu, token, ... hay những gì cần được giữ bí mật.

### Controllers

Có rất nhiều controller cho các loại dịch vụ khác nhau:

1. Deployment : là loại chung nhất, khi ta muốn "deploy" một dịch vụ nào đó. Ta tạo ra pod bằng cách tạo ra một deployment (hoặc statefulSets, hoặc các khái niệm tương đương). StatefulSets được dùng khi ta cần các service bật lên theo tứ tự nhất định.
2. DaemonSet : thường dành cho các dịch vụ cần chạy trên tất cả các node. Ví dụ như fluentd để collect log trên tất cả các node.
3. StatefulSet : là 1 file "manifest" đặt trong thư mục chỉ định bởi kubelet, các pod này sẽ được chạy khi kubelet chạy. Không thể điều khiển chúng bằng kubectl. Đây là một khái niệm đang dần bị xa lánh bởi sự thiếu linh động và khó kiểm soát.

Gõ kubectl get để xem tất cả những khái niệm resource mà k8s sử dụng, và cách gọi ngắn gọn cho từng khái niệm (svc cho service, deploy cho deployment, cm cho configmap ...).

### Helm - Trình quản lý gói của K8s

Trên Ubuntu, ta dùng APT để cài package, thì trên K8s, Helm dùng để cài các "chart” (Tương tự trình quản lý gói bên Linux). Với Helm, ta có thể triển khai các app và service như Apache Hadoop, Apache Spark, Redis, Nginx,…

### Dashboard

Dashboard cho phép xem tổng quan về cluster k8s đang dùng, nó được cài vào k8s như một add-on (link project K8s Dashboard: [https://github.com/kubernetes/dashboard](https://github.com/kubernetes/dashboard" \t "https://viblo.asia/p/_blank)) thông qua lệnh apply của kubectl. Dashboard cũng là 1 plugin có sẵn trong Minikube.

### Monitoring

Monitoring trên K8s rất dễ dàng, chỉ cần cài 1 phần mềm có khả năng tích hợp với k8s, nó sẽ hỏi K8s để lấy thông tin về tất cả các pod trong hệ thống.

### Create

Mọi file cấu hình cho pod/svc/cm/ ... đều là file ở định dạng YAML. Sau đây là 1 ví dụ về việc deploy các cấu hình từ file YAML:

kubectl create -f filename # hoặc . để cài hết các file trong thư mục hiện tại.

1. **Mô hình hệ thống được sử dụng:**

Nhóm em sử dụng dịch vụ Amazon EC2 để tạo một máy ảo chạy hệ điều hành Amazon Linux 2 và cài đặt Docker với Kubectl và minikube phiên bản mới nhất.

Nhóm em đã sử dụng Minikube để tự động hoá việc cấu hình cụm Kubernetes trên EC2 (Do tài khoản giáo dục miễn phí cho sinh viên không cho tạo secret key trong môi trường sandbox).

1. **Tài liệu hướng dẫn sử dụng:**

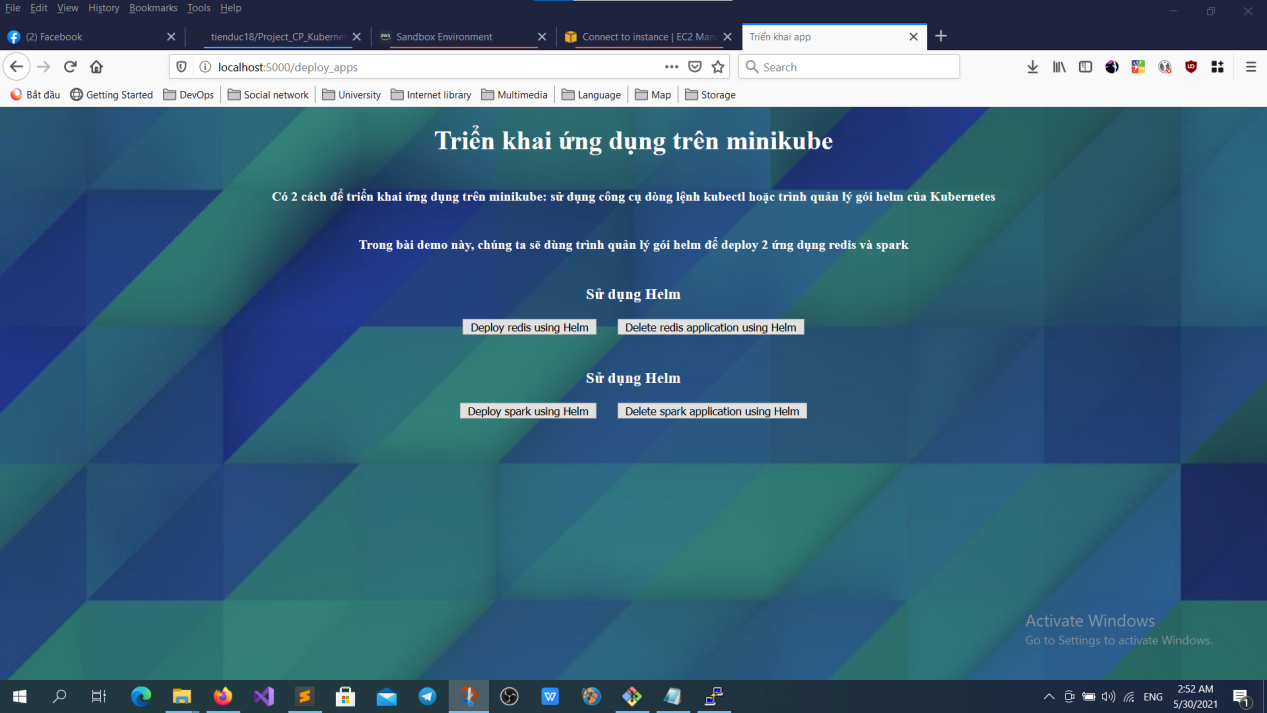
* Khởi động trang web bằng cách thực thi câu lệnh:

python index.py

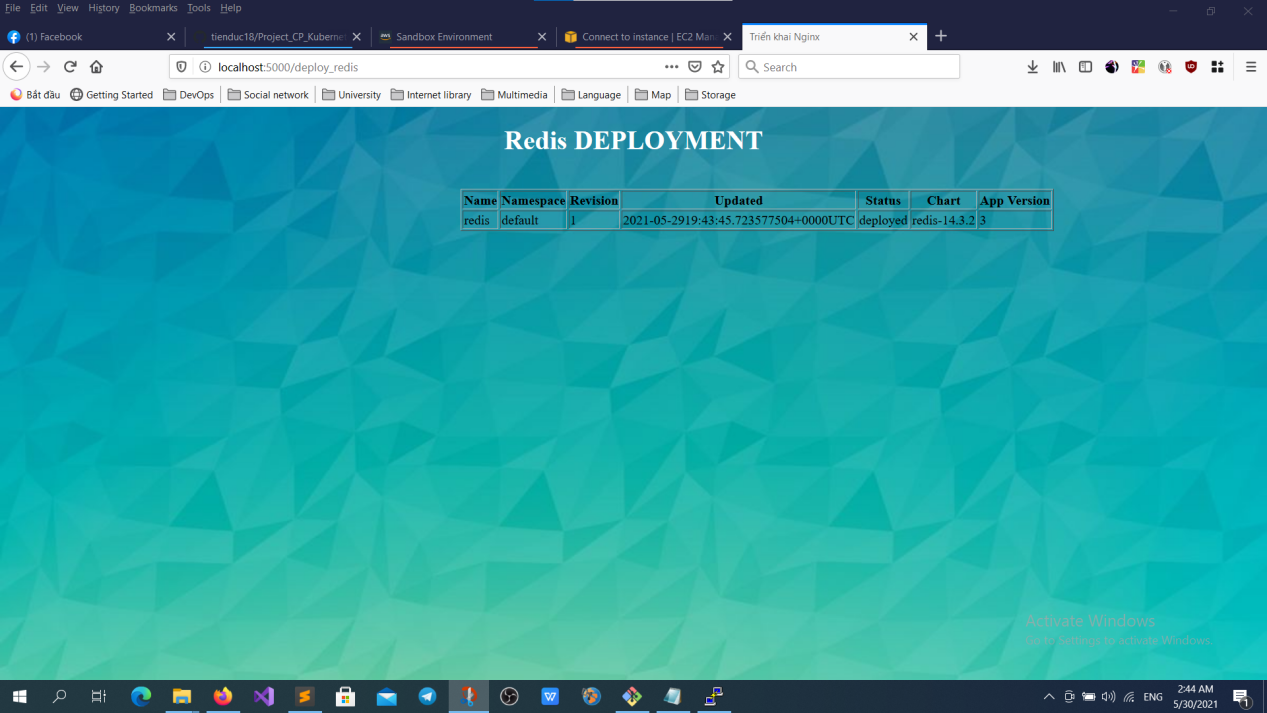
* Sau đó, trang chủ sẽ hiện ra tại cổng 5000:



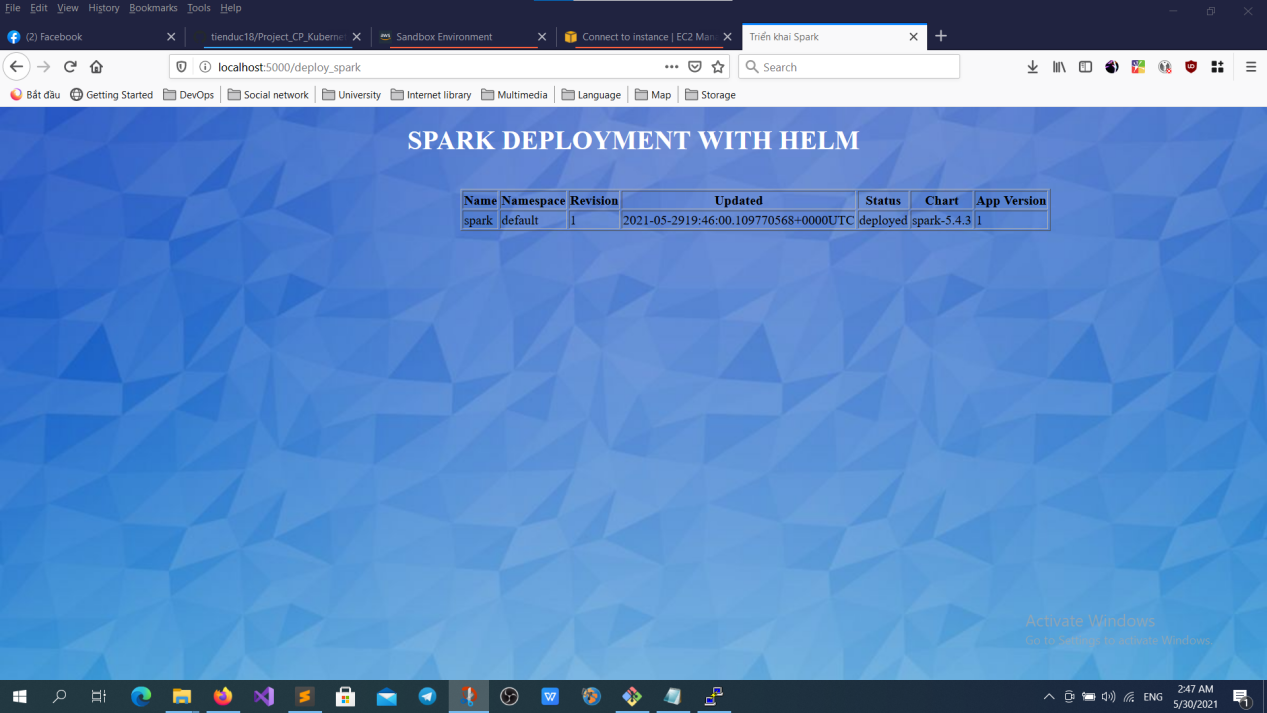
* Ở trang chủ này, ta có 3 nút để điều khiển cluster thông qua Minikube:
  + Start cluster: Khởi động cluster 1 node bằng Minikube.
  + Stop cluster: Dừng tất cả hoạt động của node trong cluster.
  + Delete cluster: Xoá bỏ toàn bộ các node và các file cấu hình trong cluster.
* Khi click vào nút “Start cluster”, trang web sẽ load 1 lúc để chuẩn bị cluster 1 node cho việc triển khai ứng dụng trên cụm và sẽ chuyển qua 1 trang lựa chọn triển khai ứng dụng như sau.



* Trang này cho phép triển khai 2 ứng dụng phổ biến ở thời điểm hiện tại là Redis và Apache Spark thông qua trình quản lý gói Helm.
* Khi click vào nút deploy với ứng dụng tương ứng, một trang web sau sẽ hiện ra:
  + Đối với Redis:



* + Đối với Spark:



* + Bảng trong hình ảnh trên cho biết trạng thái (status) của ứng dụng được cài đặt qua Helm cũng như phiên bản và namespace mà ứng dụng được triển khai.
* Trong trường hợp client muốn xoá ứng dụng thì công việc tiếp theo mà client phải làm là click vào nút “delete” tương ứng với ứng dụng đó.

1. **Vấn đề còn tồn tại:**

* Không thể sử dụng command kubeadm init để khởi tạo cluster do máy ảo Amazon Linux 2 trên EC2 không cấp quyền tạo secret key và access key mới.
* Những phương pháp khác để triển khai cụm K8s như sử dụng Rancher container để quản lý cụm K8s cũng thất bại hay dịch vụ Amazon EKS không cấp quyền truy cập cho những tài khoản giáo dục như tài khoản của nhóm em.
* Kỹ năng sử dụng Python Flask để viết web vẫn còn nhiều khuyết điểm trong việc thiết kế giao diện và backend.
* Do tình hình dịch bệnh hiện tại nên tụi em vẫn không thể tạo được cụm K8s trên máy thật Linux mà phải sử dụng Minikube để tiện trong việc nghiên cứu về K8s.

1. **Hướng phát triển:**

* Trao dồi kỹ năng viết web bằng Python Flask để có thể nhanh chóng triển khai ứng dụng web.

**NGUỒN THAM KHẢO**

<https://viblo.asia/p/phan-1-gioi-thieu-ve-kubernetes-924lJO6m5PM>

<https://viblo.asia/p/phan-2-kien-truc-cua-kubernetes-RQqKLnr6l7z#_helm---k8s-package-manager-18>

<https://kubernetes.io/vi/docs/concepts/overview/what-is-kubernetes/>