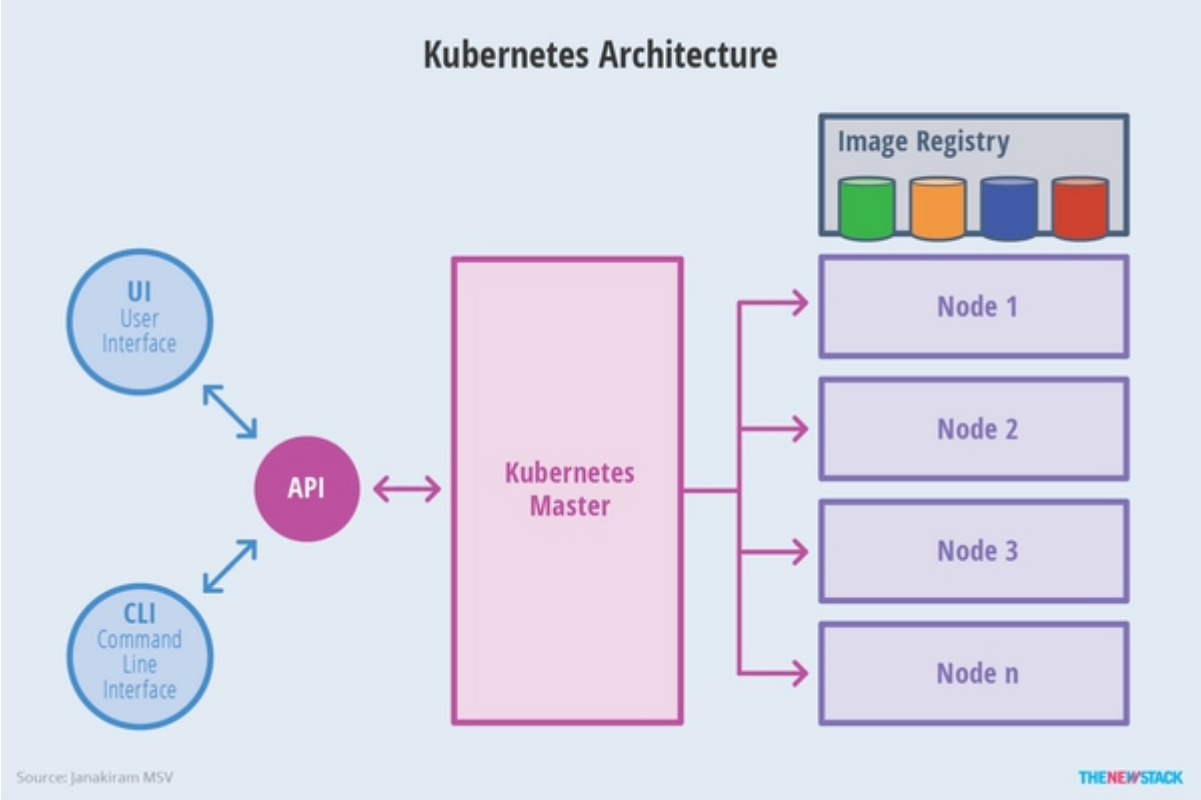
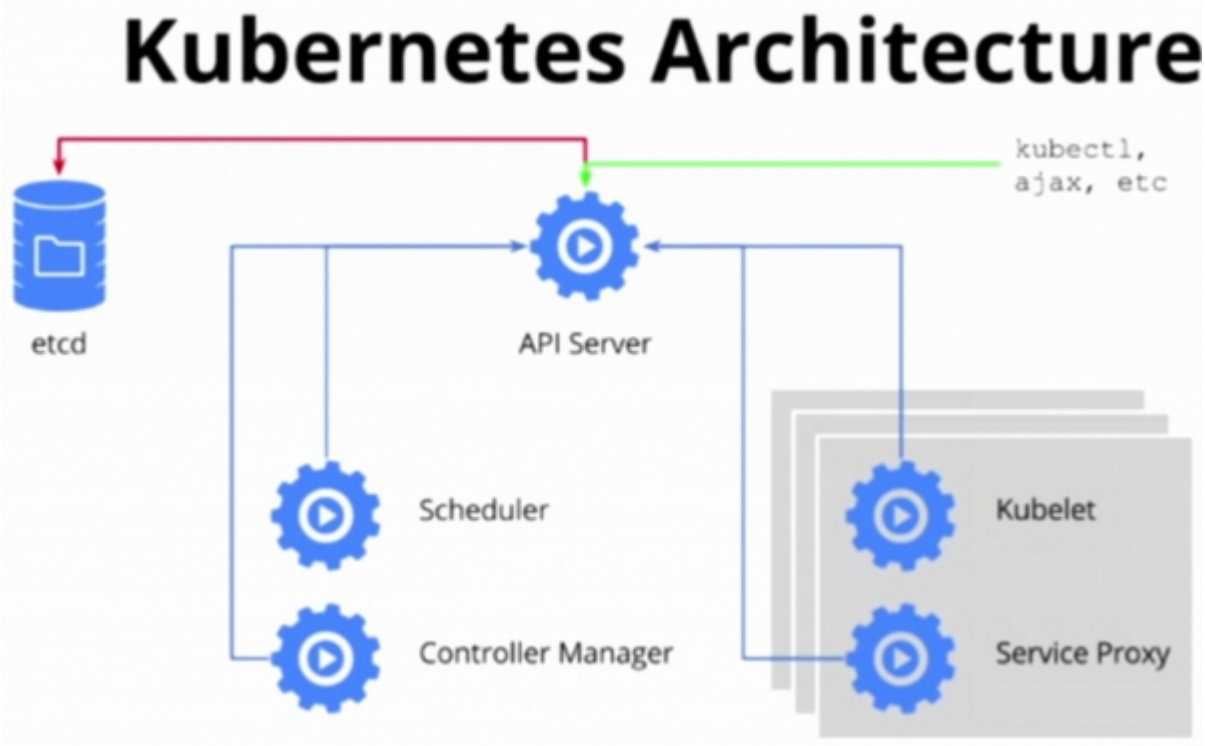
container.



1. Master server (Máy chủ)



a) Etcd

Là hệ thống lưu trữ dữ liệu của cụm K8s theo dạng key - value (Khoá - giá trị).

b) API Server

Đúng theo tên gọi, đây chính là server cung cấp Kubernetes API. 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.

c) Controller Manager Service

Chúng là các background threads chạy các task bên trong cluster. Controller bao gồm nhiều vai trò khác nhau, nhưng tất cả được compiled thành một single binary. Những vai trò của controllers bao gồm:

- **Node controller** chịu trách nhiệm cho trạng thái của worker (worker state)
- **Replication controller** chịu trách nhiệm cho việc đảm bảo duy trì (maintaining) đúng số lượng của Pods
- **End-point Controller** kết nối services và Pods với nhau.
- **Service account** và **token controllers** quản lý access management..

d) 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.

e) Dashboard (Không bắt buộc)

Giao diện web Kubernetes Dashboard giúp đơn giản hóa các tương tác của người dùng K8s thông qua API server.

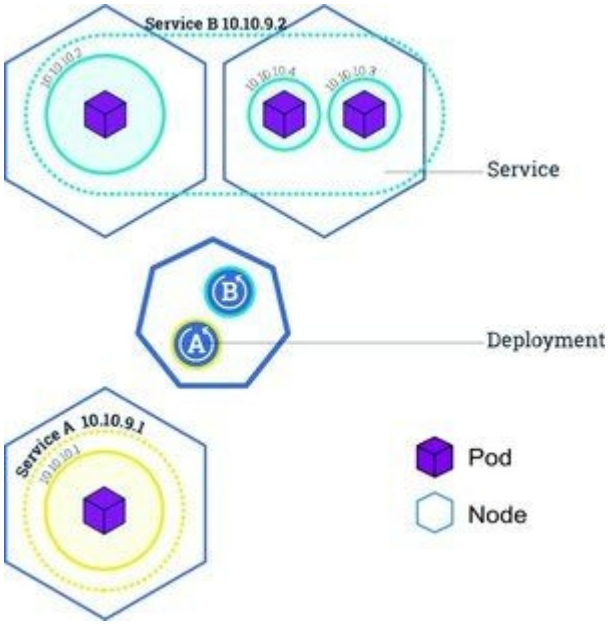
2. Node Server (Máy công nhân)

a) Pod

Pod là 1 nhóm (gồm một hoặc nhiều) container thực hiện một mục đích nào đó, như là chạy phần mềm ứng dụng 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.

b) Service (svc)

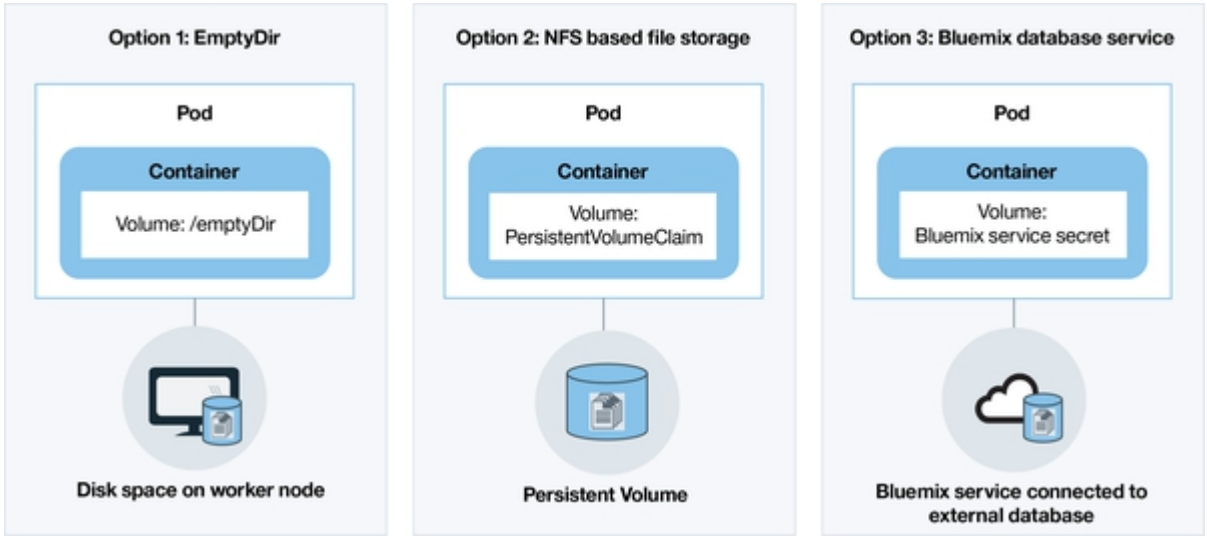
Vì các Pod có tuổi thọ ngắn, do vậy nó không đảm bảo về địa chỉ IP luôn cố định. Điều này khiến cho việc giao tiếp giữa các microservice trở nên khó khăn. Do đó, K8s giới thiệu khái niệm về svc, nó là một lớp nằm trên một số nhóm Pod. Svc cung cấp mạng máy tính đáng tin cậy bằng cách cung cấp địa chỉ IP tĩnh, DNS (Máy chủ phân giải tên miền) và cổng mạng cố định.



c) Persistent Volumes (PV)

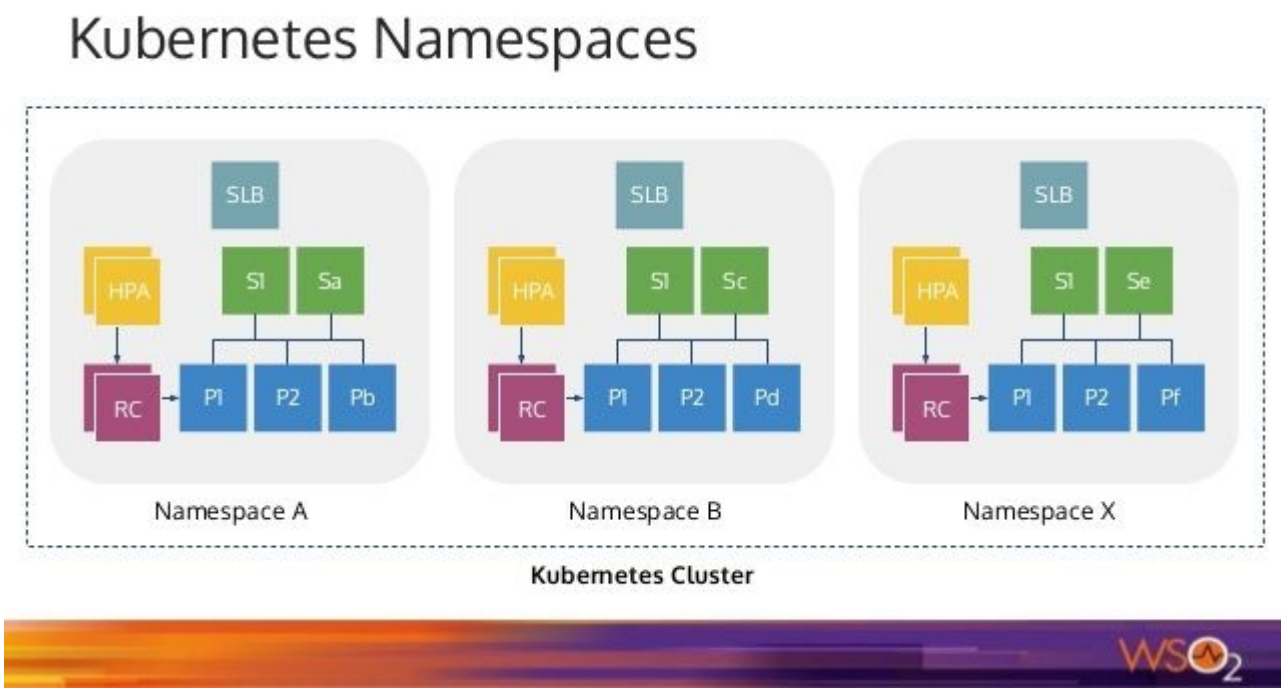
PersistentVolume (PV) là một phần không gian lưu trữ dữ liệu trong cluster, các PersistentVolume giống với Volume bình thường tuy nhiên nó tồn tại độc lập với POD (pod bị xóa PV vẫn tồn tại), có nhiều loại PersistentVolume có thể triển khai như NFS, Clusterfs …

PersistentVolumeClaim (pvc) là yêu cầu sử dụng không gian lưu trữ (sử dụng PV). Hình dung PV giống như Node, PVC giống như POD. POD chạy nó sử dụng các tài nguyên của NODE, PVC hoạt động nó sử dụng tài nguyên của PV.



d) 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.



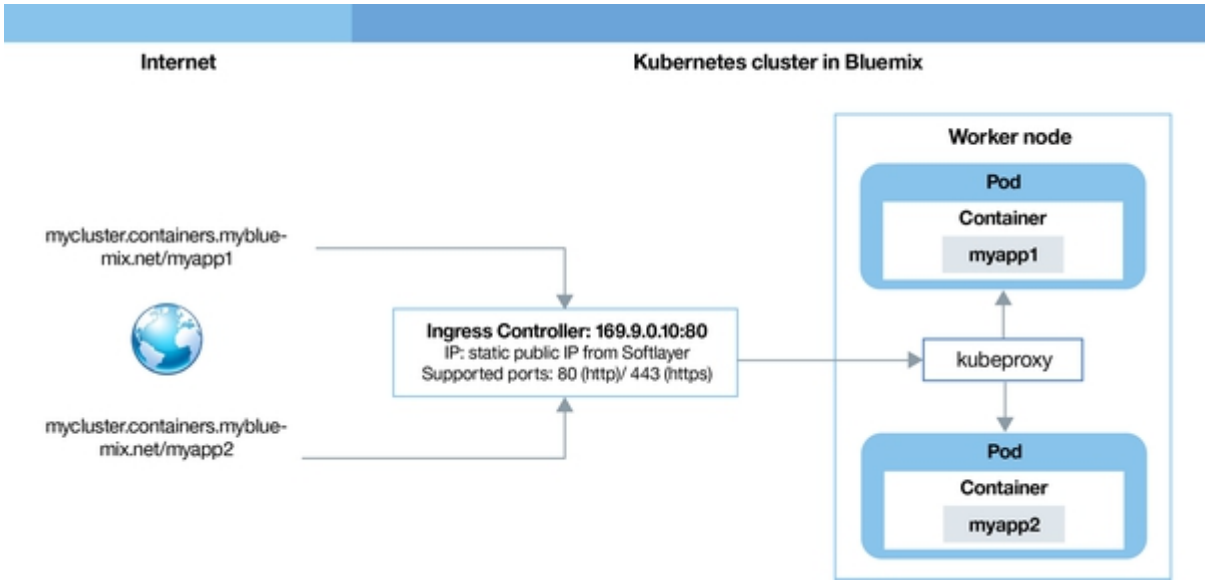
Nếu ta đặt service này là "web" lúc chạy production, còn lúc phát triển thì ta chạy nó ở đâu? Ta sẽ phải thay đổi tên service. Namespace giải quyết vấn đề này. Mặc định các dịch vụ sẽ sử đụng namespace "default", nhưng ta có thể tạo namespace tuỳ ý. K8s sử dụng 1 namespace riêng : kube-system.

e) Ingress rules

Ingress là thành phần được dùng để điều hướng các yêu cầu traffic giao thức HTTP và HTTPS từ bên ngoài (interneet) vào các dịch vụ bên trong Cluster.

Ingress chỉ để phục vụ các cổng, yêu cầu HTTP, HTTPS còn các loại cổng khác, giao thức khác để truy cập được từ bên ngoài thì dùng Service với kiểu NodePort và

LoadBalancer.



f) Network policies

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

g) 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.

h) 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 variable (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.

i) 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 ...).

j) Helm - Trình quản lý gói của K8s

Trên Ubuntu, ta có APT (Advanced Package Tool - Công cụ quản lý gói nâng cao) để cài gói phần mềm, thì trên K8s, Helm được 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,…

k) Dashboard

Dashboard cho phép xem tổng quan về cụm k8s đã được thiết lập từ trước, nó được cài vào k8s như một add-on (link project K8s Dashboard: https://github.com/kubernetes/dashboard) thông qua lệnh apply của kubectl. Dashboard cũng là 1 plugin có sẵn trong Minikube.

l) 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.

VI. Mô hình hệ thống được sử dụng:

- Nền tảng: Amazon Linux 2 (Trên Amazon EC2)
- Phiên bản Python: 3.7
- Loại máy EC2: t3.medium
- CPU: 2 nhân
- RAM: 4 GB

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).

VII. Tài liệu hướng dẫn sử dụng:

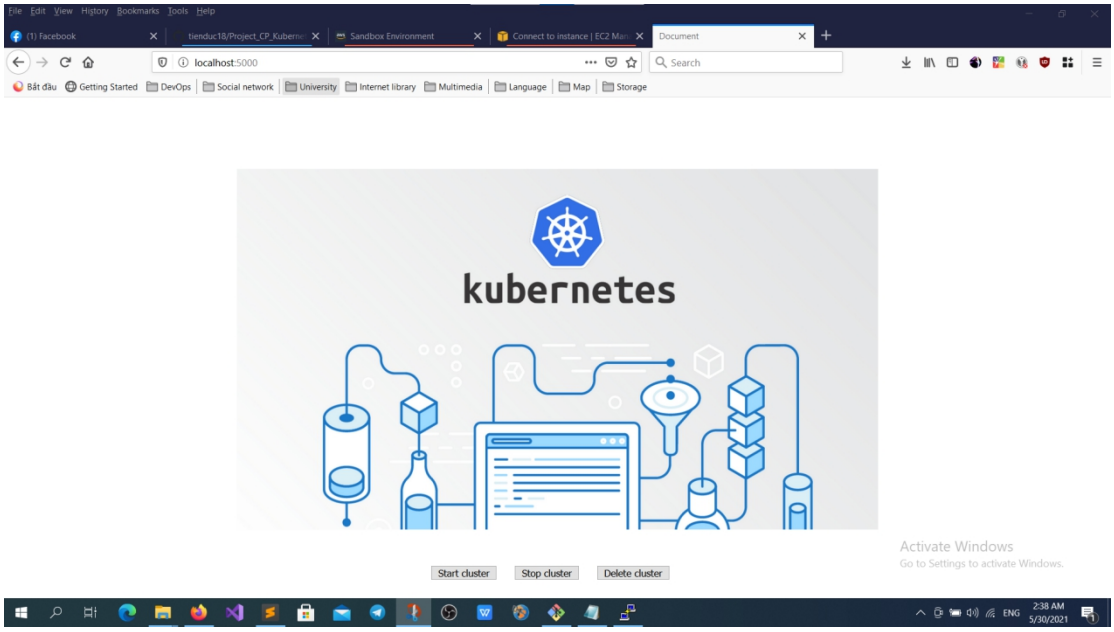
- Cài đặt Flask: pip install Flask
- Clone repo chứa web của nhóm về từ Github:

git clone https://github.com/tienduc18/Project_CP_Kubernetes.git

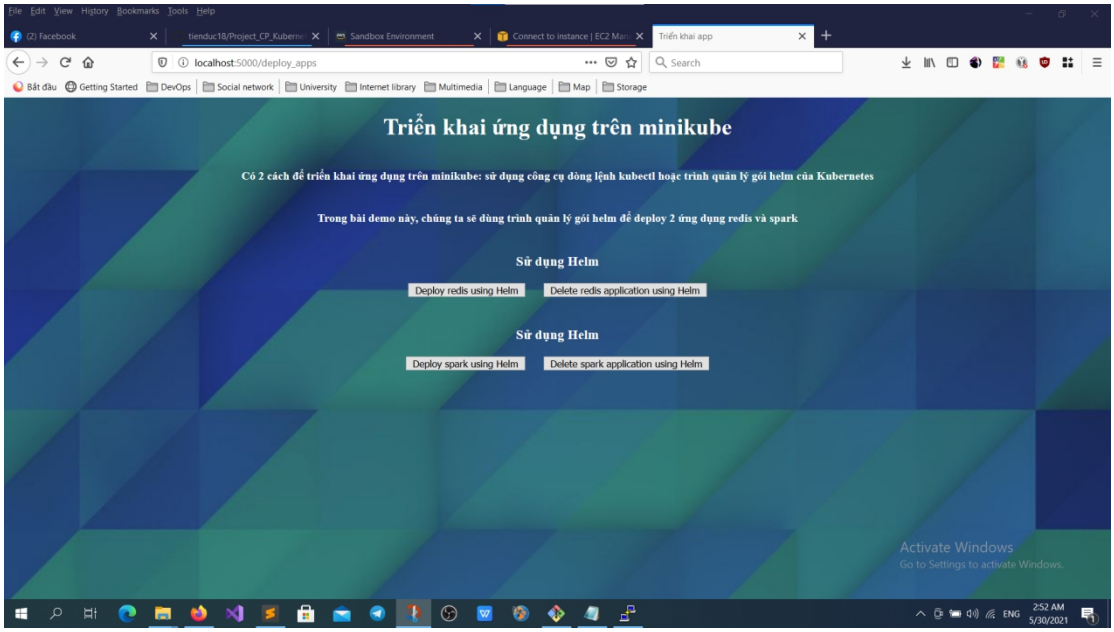
- Khởi động trang web bằng cách thực thi câu lệnh:

python index.py

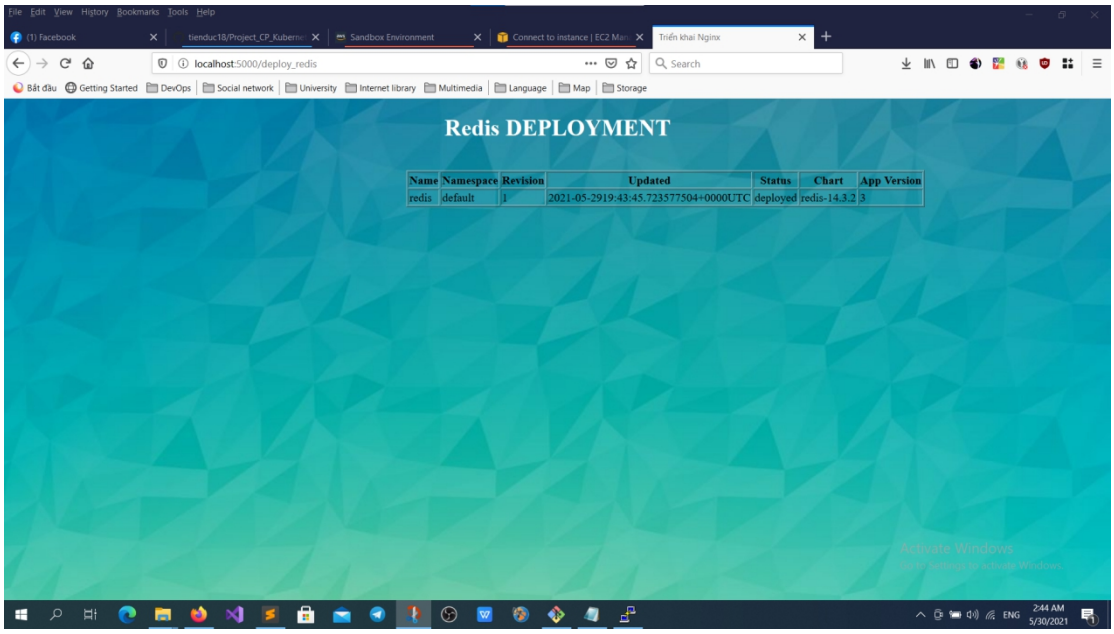
- Sau đó, trang chủ sẽ hiện ra tại cổng 5000:



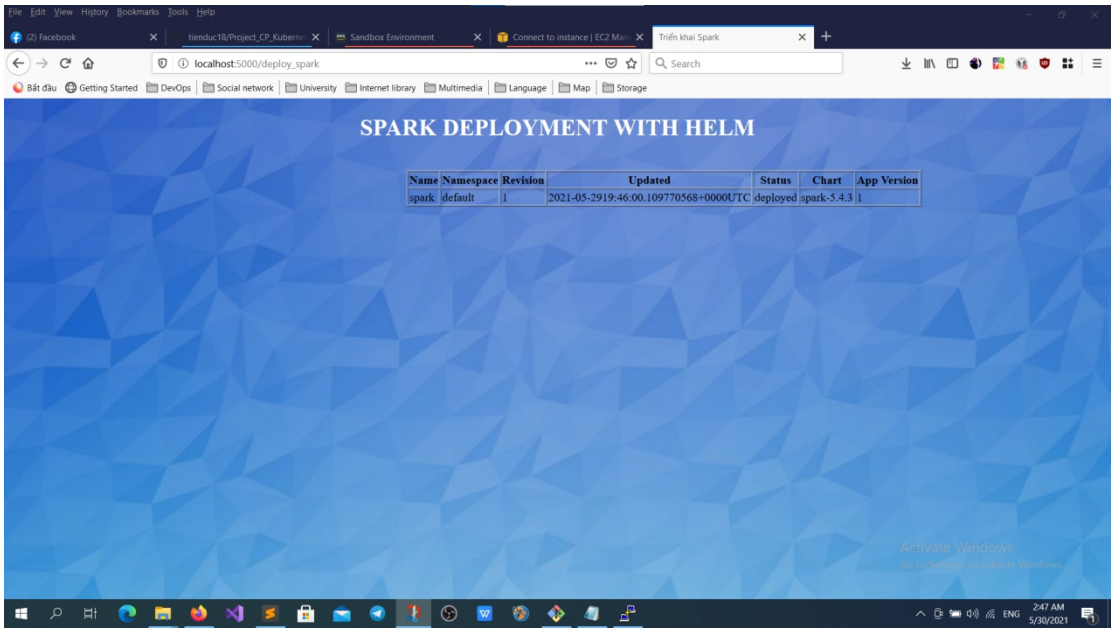
- Ở trang chủ này, ta có 3 nút để điều khiển cluster thông qua Minikube:
  - Start cluster: Khởi động cluster 1 node bằng Minikube.
  - Stop cluster: Dừng tất cả hoạt động của node trong cluster.
  - Delete cluster: Xoá bỏ toàn bộ các node và các file cấu hình trong cluster.
- Khi click vào nút "Start 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.

VIII. 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.
- Cổng mạng 8080 của EC2 từ chối không cho truy cập qua SSH và môi trường sandbox không cấp quyền truy cập EC2 Serial Console.
- 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 miễn phí trong một năm 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.

IX. Hướng phát triển:

- Ứng dụng cho ngành Kỹ thuật dữ liệu
- Kết hợp với kiến trúc Serverless của Amazon
- Tập trung vào vấn đề bảo mật cụm
- Mở rộng kiến trúc phân tán của Kubernetes để phù hợp với AI/ML
- Tự động hóa các tác vụ cài đặt và triển khai phần mềm.

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/

https://xuanthulab.net/kubernetes/

Link Github repo chứa web viết bằng Python Flask của nhóm em:

https://github.com/tienduc18/Project_CP_Kubernetes.git