**BÁO CÁO ĐỀ TÀI**

**KUBERNETES**

***MÔN HỌC: MẠNG MÁY TÍNH NÂNG CAO***

***NHÓM: B6***

***LỚP: 20HCB2***

1. **Tổng quan về Kubernetes.**
2. ***Kubernetes là gì?***

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.

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.

1. ***Kubernetes có thể làm gì?***

Các container là một cách tốt để đóng gói và chạy các ứng dụng của bạn. Trong môi trường production, bạn cần quản lý các container chạy các ứng dụng và đảm bảo rằng không có khoảng thời gian downtime. Ví dụ, nếu một container bị tắt đi, một container khác cần phải khởi động lên. Điều này sẽ dễ dàng hơn nếu được xử lý bởi một hệ thống.

Đó là cách Kubernetes đến với chúng ta. Kubernetes cung cấp cho bạn một framework để chạy các hệ phân tán một cách mạnh mẽ. Nó đảm nhiệm việc nhân rộng và chuyển đổi dự phòng cho ứng dụng của bạn, cung cấp các mẫu deployment và hơn thế nữa. Ví dụ, Kubernetes có thể dễ dàng quản lý một triển khai canary cho hệ thống của bạn.

Kubernetes cung cấp cho bạn:

* *Service discovery và cân bằng tải*Kubernetes có thể expose một container sử dụng DNS hoặc địa chỉ IP của riêng nó. Nếu lượng traffic truy cập đến một container cao, Kubernetes có thể cân bằng tải và phân phối lưu lượng mạng (network traffic) để việc triển khai được ổn định.
* *Điều phối bộ nhớ*Kubernetes cho phép bạn tự động mount một hệ thống lưu trữ mà bạn chọn, như local storages, public cloud providers, v.v.
* *Tự động rollouts và rollbacks*Bạn có thể mô tả trạng thái mong muốn cho các container được triển khai dùng Kubernetes và nó có thể thay đổi trạng thái thực tế sang trạng thái mong muốn với tần suất được kiểm soát. Ví dụ, bạn có thể tự động hoá Kubernetes để tạo mới các container cho việc triển khai của bạn, xoá các container hiện có và áp dụng tất cả các resource của chúng vào container mới.
* *Đóng gói tự động*Bạn cung cấp cho Kubernetes một cluster gồm các node mà nó có thể sử dụng để chạy các tác vụ được đóng gói (containerized task). Bạn cho Kubernetes biết mỗi container cần bao nhiêu CPU và bộ nhớ (RAM). Kubernetes có thể điều phối các container đến các node để tận dụng tốt nhất các resource của bạn.
* *Tự phục hồi*Kubernetes khởi động lại các containers bị lỗi, thay thế các container, xoá các container không phản hồi lại cấu hình health check do người dùng xác định và không cho các client biết đến chúng cho đến khi chúng sẵn sàng hoạt động.
* *Quản lý cấu hình và bảo mật*Kubernetes cho phép bạn lưu trữ và quản lý các thông tin nhạy cảm như: password, OAuth token và SSH key. Bạn có thể triển khai và cập nhật lại secret và cấu hình ứng dụng mà không cần build lại các container image và không để lộ secret trong cấu hình stack của bạn.

1. ***Những lợi ích Kubernetes mang lại***

Kubernetes được thiết kế để cung cấp developer những lợi ích sau:

* Kubernetes cung cấp các công cụ mà bạn cần để phát triển application nhanh chóng trong khi vẫn duy trì sử ổn định. Kubernetes sử dụng Container Image mà trong đó application được đóng gói. Khi bạn thêm chức năng mới, tương đương với việc tạo một Container Image mới. Khi deploy, bạn chỉ cần thay thế Image cũ bằng Image mới. Nếu có lỗi, bạn có thể roll-back ngay lập tức bằng cách deploy lại Image cũ.
* Application sẽ được chia nhỏ thành nhiều Service mà mỗi Service sẽ chỉ thực hiện một chức năng duy nhất (còn được gọi là microservice). Mỗi Service sẽ được duy trì bởi một nhóm và có thể scale dễ dàng hơn rất nhiều so với trong hệ thống thông thường.
* Kubernetes tự động khôi phục nếu có sự cố. Khi một Container dừng hoạt động, Kubernetes sẽ tự động lên lịch để chạy một Container khác.
* Nhiều application có thể chạy trên cùng một máy mà không ảnh hưởng đến nhau. Developer có thể tập trung vào việc tạo thay vì nghĩ về vị trí các ứng dụng chạy.
* Tự động hóa việc phân phối các ứng dụng trên toàn cụm, đảm bảo mức độ sử dụng cao hơn so với công cụ truyền thống.Kubernetes API giúp ứng dụng của bạn có thể di động trên nhiều môi trường khác nhau. Ví dụ, Kubernetes Services biết cách tạo bộ cân bằng tải trên tất cả các đám mây công cộng chính.

1. **Các thành phần cơ bản của Kubernetes.**
2. ***Pod***

Khi bạn triển khai ứng dụng thông qua Module [2](https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/), Kubernetes tạo ra Pod để lưu trữ phiên bản chạy của ứng dụng của bạn. Một Pod là một khái niệm trừu tượng của Kubernetes, đại diện cho một nhóm gồm một hoặc nhiều ứng dụng containers (ví dụ như Docker hoặc rkt) và một số tài nguyên được chia sẻ cho các containers đó. Những tài nguyên đó bao gồm:

* Lưu trữ được chia sẻ, dưới dạng Volumes
* Kết nối mạng, như một cluster IP duy nhất
* Thông tin về cách chạy từng container, chẳng hạn như phiên bản container image hoặc các ports cụ thể để sử dụng

Một Pod mô phỏng một "máy chủ logic" dành riêng cho ứng dụng và có thể chứa các ứng dụng containers khác nhau được liên kết tương đối chặt chẽ. Ví dụ, một Pod có thể bao gồm cả container với ứng dụng Node.js của bạn cũng như một container khác cung cấp dữ liệu hiển thị bởi webserver của Node.js. Các containers trong một Pod chia sẻ một địa chỉ IP và port space, chúng luôn được đặt cùng vị trí, cùng lên lịch trình, và chạy trong ngữ cảnh được chia sẻ trên cùng một Node.

Pods là các đơn vị nguyên tử trên nền tảng Kubernetes. Khi chúng tôi tạo một kịch bản triển khai (Deployment) trên Kubernetes, kịch bản triển khai đó tạo ra các Pods với các containers bên trong chúng (trái ngược với việc tạo các containers trực tiếp). Mỗi Pod được gắn với Node nơi nó được lên lịch trình, và tiếp tục ở đó cho đến khi chấm dứt (theo chính sách khởi động lại). Trong trường hợp có lỗi ở Node, các Pods giống nhau được lên lịch trình trên các Nodes có sẵn khác trong cluster.

1. ***ReplicaSets***

ReplicaSet là một điều khiển Controller - nó đảm bảo ổn định các nhân bản (số lượng và tình trạng của POD, replica) khi đang chạy.

ReplicaSets trong Kubernetes cũng có vai trò tương tự như ReplicationControllers, nói chính xác hơn thì ReplicaSets được giới thiệu nhằm thay thế ReplicationControllers.

Cách thức hoạt động: Khi định nghĩa một ReplicaSet (định nghĩa trong file .yaml) gồm các trường thông tin, gồm có trường selector để chọn ra các các Pod theo label, từ đó nó biết được các Pod nó cần quản lý(số lượng POD có đủ, tình trạng các POD). Trong nó nó cũng định nghĩa dữ liệu về Pod trong spec template, để nếu cần tạo Pod mới nó sẽ tạo từ template đó. Khi ReplicaSet tạo, chạy, cập nhật nó sẽ thực hiện tạo / xóa POD với số lượng cần thiết trong khai báo (repilcas).

1. ***Service***

Các POD được quản lý trong Kubernetes, trong vòng đời của nó chỉ diễn ra theo hướng - được tạo ra, chạy và khi nó kết thúc thì bị xóa và khởi tạo POD mới thay thế. ! Có nghĩa ta không thể có tạm dừng POD, chạy lại POD đang dừng ...

Mặc dù mỗi POD khi tạo ra nó có một IP để liên lạc, tuy nhiên vấn đề là mỗi khi POD thay thế thì là một IP khác, nên các dịch vụ truy cập không biết IP mới nếu ta cấu hình nó truy cập đến POD nào đó cố định. Để giải quết vấn đề này sẽ cần đến Service.

Service (micro-service) là một đối tượng trừu tượng nó xác định ra một nhóm các POD và chính sách để truy cập đến POD đó. Nhóm cá POD mà Service xác định thường dùng kỹ thuật Selector (chọn các POD thuộc về Service theo label của POD).

Cũng có thể hiểu Service là một dịch vụ mạng, tạo cơ chế cân bằng tải (load balancing) truy cập đến các điểm cuối (thường là các Pod) mà Service đó phục vụ.

1. ***Ingress***

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

Để Ingress hoặt động, hệ thồng cần một điều khiển ingress trước (Ingress controller), có nhiều loại để chọn sử dụng: [AKS Application Gateway Ingress Controller](https://azure.github.io/application-gateway-kubernetes-ingress/), [Ambassador](https://www.getambassador.io/) , [Apache APISIX ingress controller](https://github.com/apache/apisix-ingress-controller), [Avi Kubernetes Operator](https://github.com/vmware/load-balancer-and-ingress-services-for-kubernetes), [HAProxy Ingress](https://haproxy-ingress.github.io/),  [NGINX Ingress Controller for Kubernetes](https://www.nginx.com/products/nginx-ingress-controller/),…

1. ***ConfigMap***

ConfigMap là một resource trong [Kubernetes](https://cuongquach.com/tag/kubernetes) giúp bạn quản lý được file cấu hình ứng dụng, nội dung file cấu hình ứng dụng, các giá trị truyền cho lệnh command-line, port ứng dụng,… và rất nhiều thứ khác có thể hỗ trợ khi bạn khai báo các thành phần của 1 Pod.

ConfigMap giúp ứng dụng của bạn tách biệt phần code ứng dụng với nội dung cấu hình về hệ thống. Ví dụ như bạn dễ dàng thay đổi cấu hình về môi trường Dev/QA/UAT/Production như database connection string, domain, hostname, url,… mà không cần phải can thiệp vào code. Một trong những điều quan trọng của 12Factor (https://12factor.net/vi/config).

[ConfigMap](https://cuongquach.com/tag/configmap) rất hữu ích để lưu trữ và chia sẻ thông tin cấu hình không nhạy cảm, không được mã hóa. Để sử dụng thông tin nhạy cảm trong cụm của bạn, bạn phải sử dụng Bí mật.

1. ***Secret***

Kubernetes Secrets cho phép bạn lưu trữ và quản lý thông tin nhạy cảm, chẳng hạn như mật khẩu, mã thông báo OAuth và khóa ssh. Lưu trữ thông tin bí mật trong Secrets an toàn và linh hoạt hơn so với việc đưa nó nguyên văn vào định nghĩa Pod hoặc trong hình ảnh vùng chứa.

Secret là một đối tượng có chứa một lượng nhỏ dữ liệu nhạy cảm như mật khẩu, mã thông báo hoặc khóa. Nếu không thì thông tin đó có thể được đưa vào đặc tả Pod hoặc trong hình ảnh. Người dùng có thể tạo Secrets và hệ thống cũng tạo một số Secrets.

Để sử dụng Secrets, một Pod cần tham chiếu Secrets. Secrets có thể được sử dụng với Pod theo ba cách:

* Dưới dạng các tệp trong một ổ đĩa được gắn trên một hoặc nhiều vùng chứa của nó.
* Như biến môi trường vùng chứa.
* Bằng kubelet khi kéo images cho Pod.

1. ***Volumes***

Volumes là thành phần trực thuộc Pods. Volumes được định nghĩa trong cấu hình file yaml khi khởi tạo các Pods. Các container có thể thực hiện mount dữ liệu bên trong container đến đối tượng volumes thuộc cùng Pods.

Các loại volumes:

* emptyDir
* hostPath
* gitRepo
* nfs
* Cloud volumes: gcePersistentDisk (Google Compute Engine Persistent Disk), awsElasticBlockStore (Amazon Web Services Elastic Block Store Volume), azureDisk (Microsoft Azure Disk Volume).
* cinder, cephfs, iscsi, flocker, glusterfs, quobyte, rbd, flexVolume, vsphereVolume, photonPersistentDisk, scaleIO
* configMap, secret, downwardAPI
* persistentVolumeClaim

1. ***Deployments***

Deployment quản lý một nhóm các Pod - các Pod được nhân bản, nó tự động thay thế các Pod bị lỗi, không phản hồi bằng pod mới nó tạo ra. Như vậy, deployment đảm bảo ứng dụng của bạn có một (hay nhiều) Pod để phục vụ các yêu cầu.

Deployment sử dụng mẫu Pod (Pod template - chứa định nghĩa / thiết lập về Pod) để tạo các Pod (các nhân bản replica), khi template này thay đổi, các Pod mới sẽ được tạo để thay thế Pod cũ ngay lập tức.

Có nhiều kịch bản sử dụng Deployment:

* Phát hành các bản cập nhật mới.
* Khai báo các trạng thái mới của các Pods bằng cách update PodTemplateSpec của Deployment.
* Rollback về bản phát hành trước đó nếu bản mới xảy ra vấn đề.
* Scale Deployment để đáp ứng tải nhiều hơn.
* Tạm dừng Deployment để áp dụng các bản vá PodTemplateSpec, sau đó tiếp tục triển khai để phát hành bản mới.
* Sử dụng status của Deployment để xác định được khi nào thì một phát hành bị dừng lại/xảy ra lỗi.
* Dọn dẹp các bản ReplicaSets cũ mà không còn sử dụng.

1. **Kiến trúc của Kubernetes.**
2. ***Node***

Một Pod luôn chạy trên một Node. Một Node là một máy worker trong Kubernetes và có thể là máy ảo hoặc máy vật lý, tuỳ thuộc vào cluster. Mỗi Node được quản lí bởi Master. Một Node có thể chứa nhiều Pods và Kubernetes master tự động xử lí việc lên lịch trình các Pods thuộc các Nodes ở trong cluster. Việc lên lịch trình tự động của Master sẽ tính đến các tài nguyên có sẵn trên mỗi Node.

Mỗi Node ở Kubernetes chạy ít nhất:

* Kubelet, một quy trình chịu trách nhiệm liên lạc giữa Kubernetes Master và Node; quản lí các Pods và các containers đang chạy trên cùng một máy.
* Một container runtime (như Docker, rkt) chịu trách nhiệm lấy container image từ registry, giải nén container và chạy ứng dụng. Các containers chỉ nên được lên lịch trình cùng nhau trong một Pod duy nhất nếu chúng được liên kết chặt chẽ.

1. ***Master node***

Master node Là máy chủ điều khiển các máy Worker chạy ứng dụng. Master node bao gồm 4 thành phần chính:

* API Server: là thành phần giúp các thành phần khác liên lạc nói chuyện với nhau, khi triển khai ứng dụng sẽ gọi API Kubernetes API Server này.
* Etcd:  là nơi lưu trữ toàn bộ dữ liệu của Kubernetes, tất cả các thông tin của Kubernetes được lưu trữ cố định vào đây.
* Scheduler: thành phần này lập lịch triển khai cho các ứng dụng được đặt vào Worker nào để chạy
* Controller Manager: thành phần đảm nhiệm phần quản lý các Worker, kiểm tra các Worker sống hay chết, đảm nhận việc nhân bản ứng dụng…

1. ***Worker node***

Worker node là máy chủ chạy ứng dụng trên đó. Bao gồm 3 thành phần chính:

* Kubelet: là thành phần giao tiếp với Kubernetes API Server, và cũng quản lý các container.
* Kube-proxy: Thành phần này đảm nhận việc phân tải giữa các ứng dụng.
* Container runtime: là thành phần giúp chạy các ứng dụng dưới dạng Container. Thông thường người ta sử dụng Docker.

1. **Cài đặt Kubernetes (Windows)**

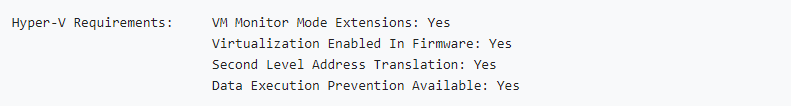
***Minikube***

*Trước khi bắt đầu*

Để kiểm tra xem việc ảo hóa (virtualization) có được hỗ trợ trên Windows 8 và các phiên bản Windows cao hơn không, chạy lệnh sau trên terminal của Windows hoặc command promt.



Nếu bạn thấy những thông tin sau, ảo hóa được hỗ trợ trên Windows.



Nếu bạn thấy thông tin sau, thì hệ thống đã được cài đặt Hypervisor và bạn có thể bỏ qua bước tiếp theo.



*Cài đặt minikube*

Cài đặt kubectl

Đảm bảo bạn đã cài đặt kubectl.

Cài đặt Hypervisor

Nếu bạn chưa cài đặt Hypervisor, hãy cài đặt một trong những phần mềm sau đây:

• [Hyper-V](https://msdn.microsoft.com/en-us/virtualization/hyperv_on_windows/quick_start/walkthrough_install)

• [VirtualBox](https://www.virtualbox.org/wiki/Downloads)

**Ghi chú:** Hyper-V có thể chạy trên 3 phiên bản khác nhau của Windows 10: Windows 10 Enterprise, Windows 10 Professional, và Windows 10 Education.

Cài đặt Minikube sử dụng Chocolatey

Cách đơn giản nhất để cài đặt Minikube trên Windows là sử dụng [Chocolatey](https://chocolatey.org/) (chạy với quyền admin):



Sau khi Minikube hoàn tất việc cài đặt, hãy đóng CLI hiện tại và khởi động lại. Minikube sẽ được tự động thêm vào biến môi trường path của bạn.

Cài đặt Minikube sử dụng gói cài đặt thực thi

Để cài đặt Minikube thủ công trên Windows sử dụng [Windows Installer](https://docs.microsoft.com/en-us/windows/desktop/msi/windows-installer-portal), tải về [minikube-installer.exe](https://github.com/kubernetes/minikube/releases/latest/download/minikube-installer.exe) và chạy bản cài đặt đó.

Cài đặt Minikube thông qua tải về trực tiếp

Để cài đặt Minikube thủ công trên Windows, tải về [minikube-windows-amd64](https://github.com/kubernetes/minikube/releases/latest), đổi tên nó thành minikube.exe, và thêm nó vào biến môi trường path.

*Dọn dẹp local state*

Nếu bạn đã cài Minikube trước đó, và chạy:



và tiếp đó minikube start trả về lỗi:



thì tiếp theo bạn cần xóa bỏ local state của minikube:



***Kubectl***

Tải về phiên bản mới nhất v1.17.0 từ [đường dẫn này](https://storage.googleapis.com/kubernetes-release/release/v1.17.0/bin/windows/amd64/kubectl.exe).Hoặc nếu bạn đã cài đặt curl, hãy sử dụng câu lệnh sau:



Để tìm ra phiên bản ổn định mới nhất, hãy xem <https://storage.googleapis.com/kubernetes-release/release/stable.txt>.

Đưa bản binary vào biến môi trường PATH của bạn.

Kiểm tra chắn chắn phiên bản kubectl giống với bản đã tải về:



**Ghi chú:**[Docker Desktop cho Windows](https://docs.docker.com/docker-for-windows/#kubernetes) thêm phiên bản kubectl riêng của nó vào PATH. Nếu bạn đã cài đặt Docker Desktop trước đây, bạn có thể cần đặt đường dẫn PATH của bạn trước khi bản cài đặt Docker Desktop thêm 1 PATH vào hoặc loại bỏ kubectl của Docker Desktop.

1. **Tài liệu tham khảo**

*https://kubernetes.io*

*https://minikube.sigs.k8s.io*