

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

MÔN ĐIỆN TOÁN Đám Mây



TÌM HIỂU KUBERNETES VÀ VIẾT ỨNG DỤNG DEMO

GVHD: TS. Huỳnh Xuân Phụng

Lớp: Nhóm 03CLC_Tối thứ 3

Thực hiện:

Trần Tiến Phát 19110261

Nguyễn Văn Hoàng 19110209

Tp. Hồ Chí Minh, ngày 10 tháng 12 năm 2021

ĐIỂM SỐ

TIÊU CHÍ			TỔNG
ĐIỂM			

NHẬN XÉT CỦA GIÁO VIÊN:

TP. Hồ Chí Minh, ngày ..., tháng ..., năm 2021

Giảng viên chấm điểm

(Ký và ghi rõ họ tên)

Huỳnh Xuân Phụng

MÔ TẢ ĐỀ TÀI ĐỒ ÁN MÔN HỌC

Họ và Tên sinh viên:

Trần Tiến Phát 19110261

Nguyễn Văn Hoàng 19110209

Thời gian thực hiện : Từ : 6/9/2021 Đến : 10 /12/2021

Ngành: Công nghệ thông tin

Tên đề tài : Tìm hiểu Kubernetes và viết ứng dụng demo

GVHD: TS. Huỳnh Xuân Phụng

Nhiệm vụ của đề tài :

1. Tìm hiểu Kubernetes
2. Viết và chạy ứng dụng demo

LỜI CẢM ƠN

Lời đầu tiên, xin chân thành cảm ơn quý Thầy, Cô Trường Đại học Sư phạm Kỹ thuật TP.HCM đã giảng dạy và trang bị cho chúng tôi những kiến thức cần thiết trong suốt thời gian qua. Đặc biệt, xin được gửi lời cảm ơn sâu sắc đến thầy **TS. Huỳnh Xuân Phụng** đã tận tình truyền đạt những kiến thức quý báu cũng như hướng dẫn chúng tôi trong việc hoàn thành tiểu luận nghiên cứu. Và kiến thức là tồn tại mãi mãi nên nó sẽ là hành trang để chúng tôi tiếp tục vững vàng bước đi trên con đường học tập và lao động sau này.

Đề tài và bài báo cáo được chúng em thực hiện trong khoảng thời gian ngắn, với những kiến thức còn hạn chế cùng nhiều hạn chế khác về mặt kỹ thuật và kinh nghiệm trong việc thực hiện một dự án phần mềm. Do đó, trong quá trình làm nên đề tài có những thiếu sót là điều không thể tránh khỏi nên chúng em rất mong nhận được những ý kiến đóng góp quý báu của các quý thầy cô để kiến thức của chúng em được hoàn thiện hơn và chúng em có thể làm tốt hơn nữa trong những lần sau.

Chúng em xin chân thành cảm ơn. Cuối lời, chúng em kính chúc quý thầy, quý cô luôn dồi dào sức khỏe và thành công hơn nữa trong sự nghiệp trồng người. Một lần nữa chúng em xin chân thành cảm ơn.

PHẦN 1: TÌM HIỂU KUBERNETES.....	6
I. Kubernetes.....	6
1. Kubernetes (K8s) là gì?.....	6
2. Nên sử dụng Kubernetes khi nào?.....	6
3. Kubernetes giải quyết vấn đề gì?.....	6
4. Lợi ích của Kubernetes.....	7
II. Các dịch vụ cung cấp Kubernetes Cluster cài đặt sẵn:.....	8
III. Các cách để cài đặt K8s.....	8
IV. Các đối thủ cạnh tranh.....	8
V. Kiến trúc.....	8
1. Master server.....	9
1.1 Etcd.....	9
1.2 API Server.....	10
1.3 Controller Manager Service.....	10
1.4 Scheduler Service.....	10
1.5 Dashboard (optional).....	10
2. Node Server.....	10
2.1 Pod.....	10
2.2 Service (svc).....	11
2.3 Persistent Volumes.....	11
2.4 Namespaces.....	12
2.5 Ingress rules.....	13
2.6 Network policies.....	13
2.7 Network.....	14
2.8 ConfigMaps and Secrets.....	14
2.9 Controllers.....	14
2.10 Helm - K8s package manager.....	14
2.11 Dashboard.....	14
2.12 Monitoring.....	15
2.13 Create.....	15
PHẦN 2: VIẾT VÀ CHẠY ỨNG DỤNG DEMO.....	16
1. Tạo cụm Kubernetes.....	16
1.1 Mô hình.....	17
1.2 Tạo Cluster.....	19
2. Triển khai web NGINX trên Clusters vừa tạo.....	21
PHẦN 3: KẾT LUẬN.....	26
1. Nội dung đã thực hiện được.....	26
2. Vấn đề còn tồn đọng.....	26
3. Hướng phát triển.....	26
PHẦN 4 : TÀI LIỆU THAM KHẢO.....	27
1. Lý thuyết.....	27
2. Thực hành.....	27

PHẦN 1: TÌM HIỂU KUBERNETES

I. Kubernetes

1. Kubernetes (K8s) là gì?

Kubernetes là một hệ thống điều phối container mã nguồn mở nổi tiếng hiện nay và được sử dụng để đóng gói các service từ đó triển khai hệ thống microservices (Hệ thống mà ở đó các dịch vụ được đóng gói cách ly với nhau ở cấp độ hệ điều hành và được liên kết với nhau trong một hệ thống mạng máy tính chung). 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, kết hợp với các ý tưởng và thực tiễn tốt nhất từ cộng đồng.

2. Nên sử dụng Kubernetes khi nào?

Các doanh nghiệp lớn, có nhu cầu thực sự phải scaling hệ thống nhanh chóng, và đã sử dụng container (Docker).

Các dự án cần chạy ≥ 5 container cùng loại cho 1 dịch vụ. (Ví dụ dùng ≥ 5 máy cùng để chạy code website TopDev).

Các startup tân tiến, chịu đầu tư vào công nghệ để dễ dàng auto scale về sau.

3. Kubernetes giải quyết vấn đề gì?

Bằng việc sử dụng docker, trên 1 host bạn có thể tạo ra nhiều container. Tuy nhiên nếu bạn có ý định sử dụng trên môi trường production thì phải bắt buộc phải nghĩ đến những vấn đề dưới đây:

- Việc quản lý hàng loạt docker host
- Container Scheduling
- Rolling update
- Scaling/Auto Scaling
- Monitor vòng đời và tình trạng sống chết của container.
- Self-healing trong trường hợp có lỗi xảy ra. (Có khả năng phát hiện và tự correct lỗi)
- Service discovery

- Load balancing
- Quản lý data, work node, log
- Infrastructure as Code
- Sự liên kết và mở rộng với các hệ thống khác

Bằng việc sử dụng một Container orchestration engine như K8s có thể giải quyết được nhưng vấn đề trên đây. Trong trường hợp không sử dụng k8s, Thì sẽ phải cần thiết tạo ra cơ chế tự động hoá cho những cái kể trên, như thế thì cực kỳ tốn thời gian và không khả thi.

4. Lợi ích của Kubernetes

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

II. Các dịch vụ cung cấp Kubernetes Cluster cài đặt sẵn:

IBM: IBM Cloud Kubernetes Service

Ionos: Ionos Manager Kubernetes

Nutanix: Nutanix Karbon

OpenNebula: OpenNebula Kubernetes

OpenShift: OpenShift Dedicated and OpenShift Online

Google: Google Kubernetes Engine (GKE)

Microsoft Azure: Azure Kubernetes Service (AKS)

Amazon: Amazon EKS

III. 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 (Infrastructure as A Service -Kiến trúc hạ tầng như một dịch vụ) như OpenStack. Vậy là người ta có hàngchục cách khác nhau để cài K8s, đáng kể nhất có:

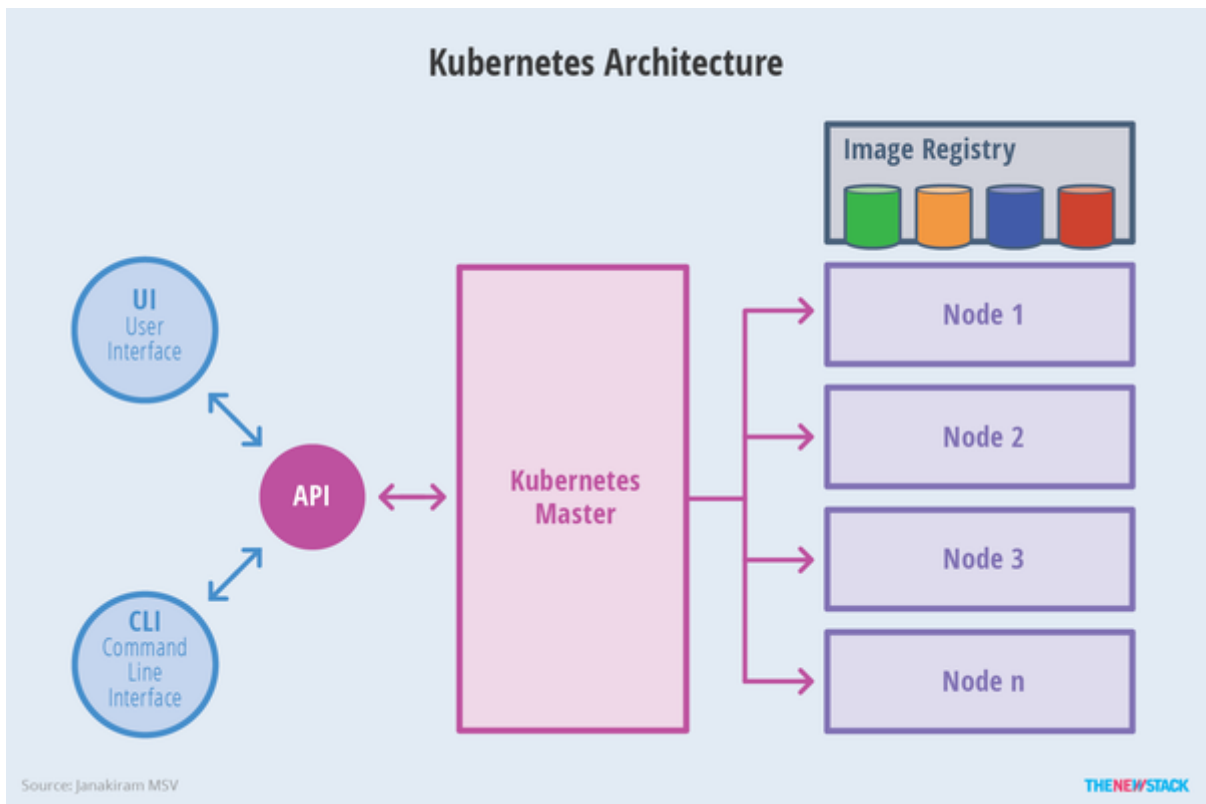
1. Minikube (Tích hợp sẵn trên Docker Desktop)
2. Kubeadm đang trong giai đoạn phát triển, để cài trên hệ thống máy vật lý máy ảo dùng Ubuntu 16.04 hay CentOS 7
3. Kargo là phần mềm dựa trên Ansible (Phần mềm tự động hoá cấu hình) để cài trên rất nhiều nơi bao gồm cả máy vật lý máy ảo/AWS/GCE
4. Dùng Salt Stack
5. Cài bằng tay trên CoreOS Container Linux
6. Kops để cài trên AWS (Amazon cloud)

IV. Các đối thủ cạnh tranh

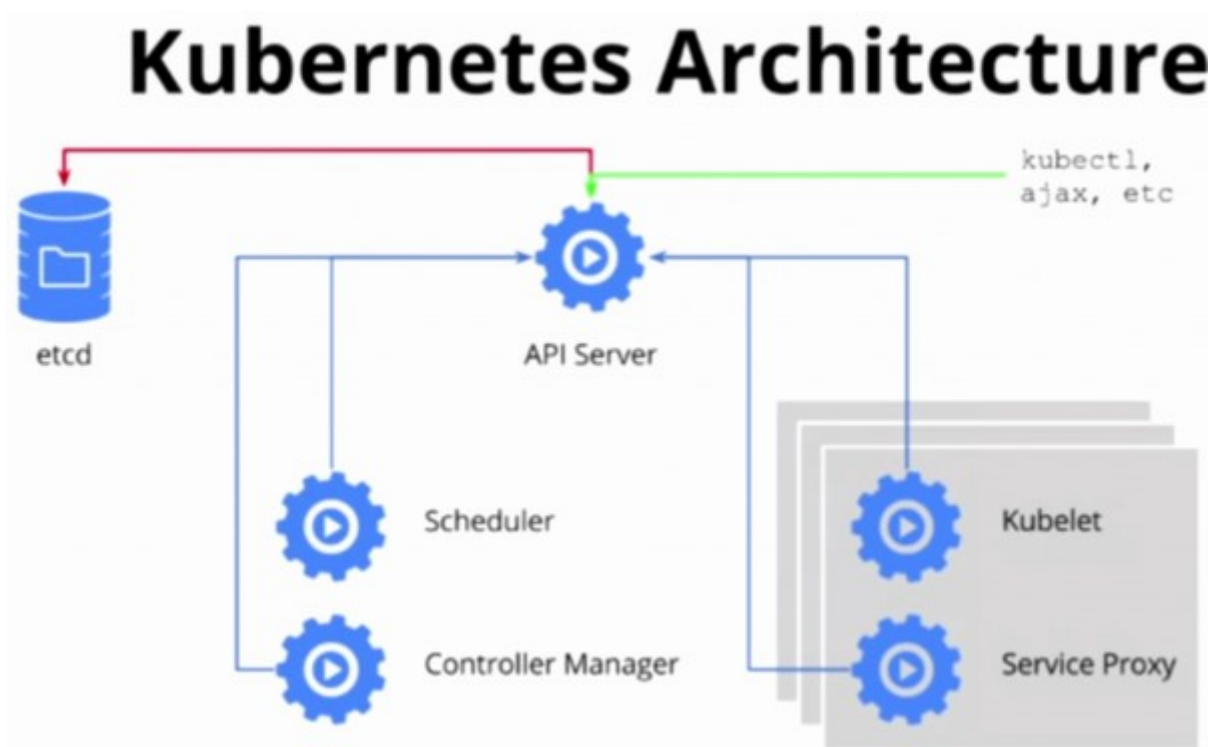
1. Docker Swarm <https://docs.docker.com/engine/swarm/>
2. Apache Mesos <https://mesos.apache.org/>

V. Kiến trúc

K8s cluster bao gồm nhiều node, trên mỗi node sẽ cần chạy một "kubernetes", 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.



1. Master server



1.1 Etcd

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

1.2 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.

1.3 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.

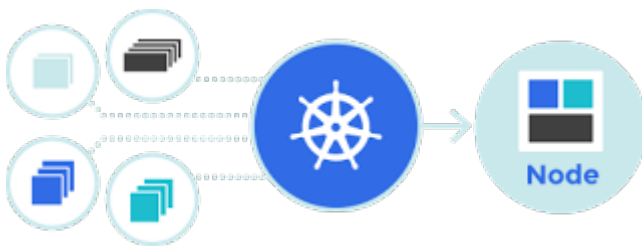
1.4 Scheduler Service

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.

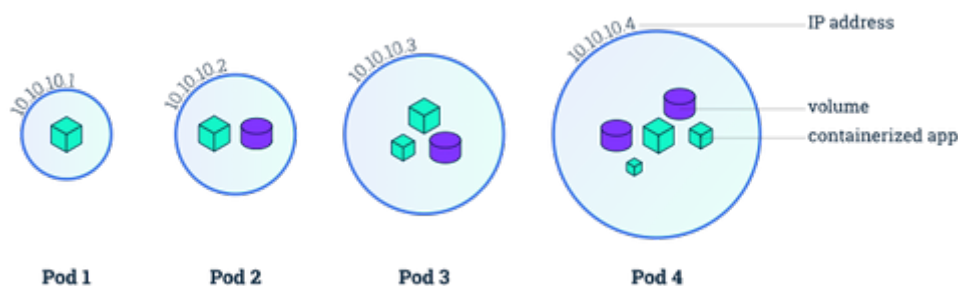
1.5 Dashboard (optional)

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

2. Node Server

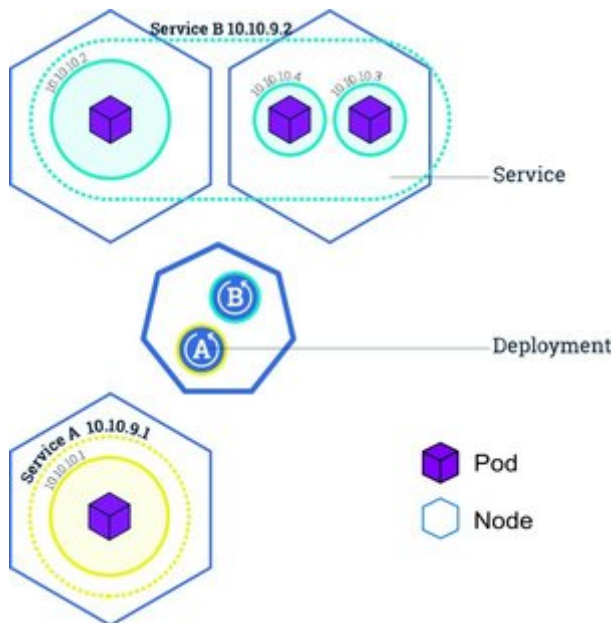


2.1 Pod



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. Nếu k8s chỉ có mỗi khái niệm pod, thì dùng k8s giống như dùng docker bình thường. Tức muốn thêm tính năng gì thì ta phải tự kiến trúc/ thiết kế/ thực hiện.

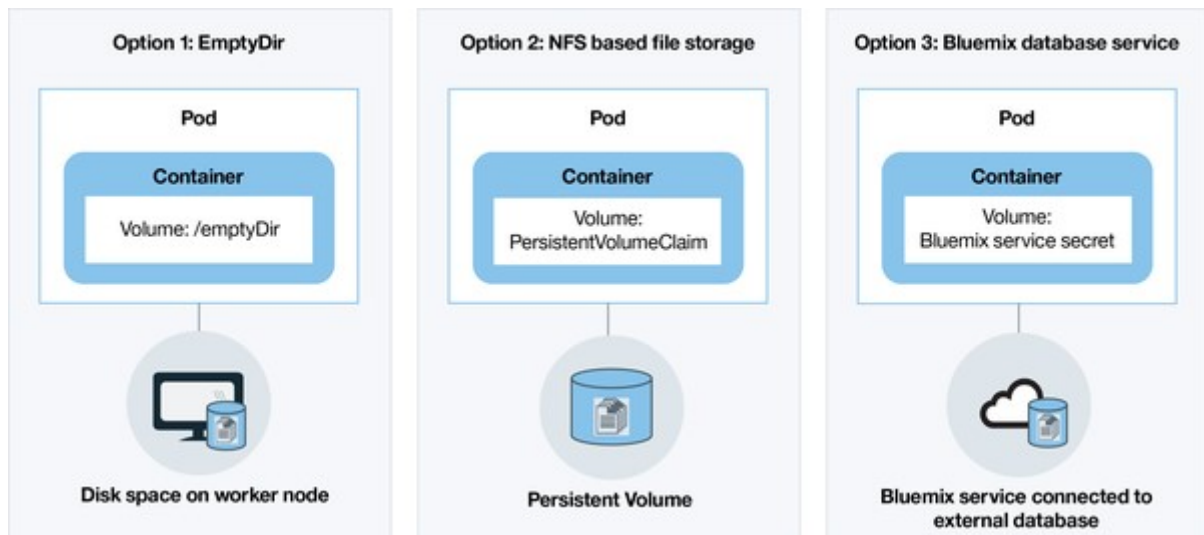
2.2 Service (svc)



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 gán 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.

Tất nhiên, nếu chỉ có 1 máy chạy 1 dịch vụ, thì service chả có nghĩa lý gì. Vậy nên khi dùng k8s, hãy nhớ rằng nó được thiết kế và đưa vào các khái niệm để phục vụ cho hàng trăm, ngàn service/container, chứ không phải 1 cái. Nó phức tạp vì nó có lý do để phức tạp. Và bạn/công ty của bạn không phải Google (ở đây không hạ thấp bạn hay công ty của bạn), không phải công ty nào cũng chạy dịch vụ software cung cấp cho cả thế giới.

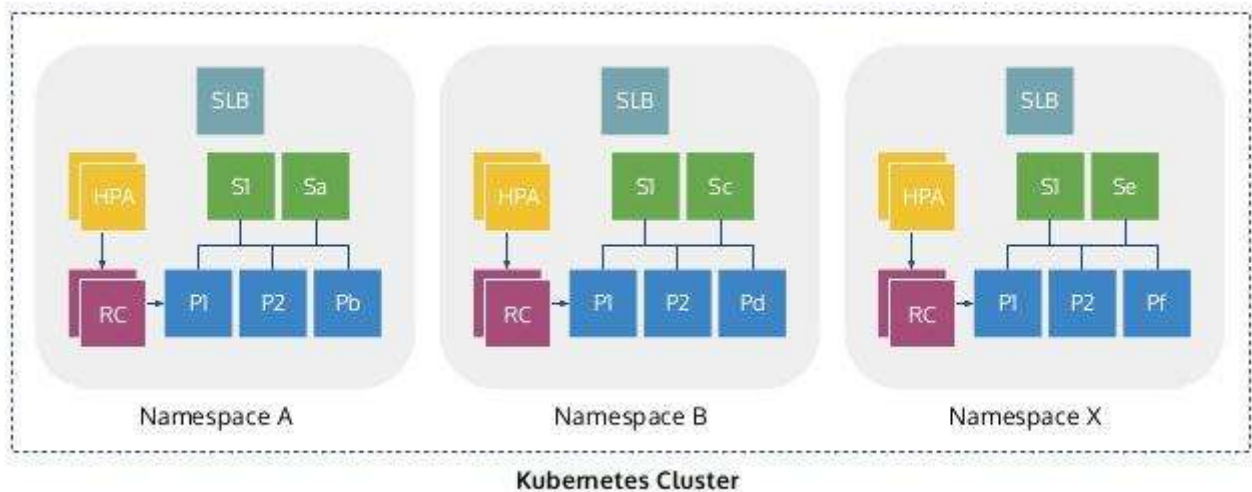
2.3 Persistent Volumes



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 restart / bị die thì dữ liệu cũng sẽ mất theo nó. Đây là chuyện dù dùng docker trực tiếp hay giải pháp khác K8s thì bạn 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 nhau. Nếu tự cài K8s, hãy chuẩn bị sẵn giải pháp lưu trữ của bạn. 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, ...

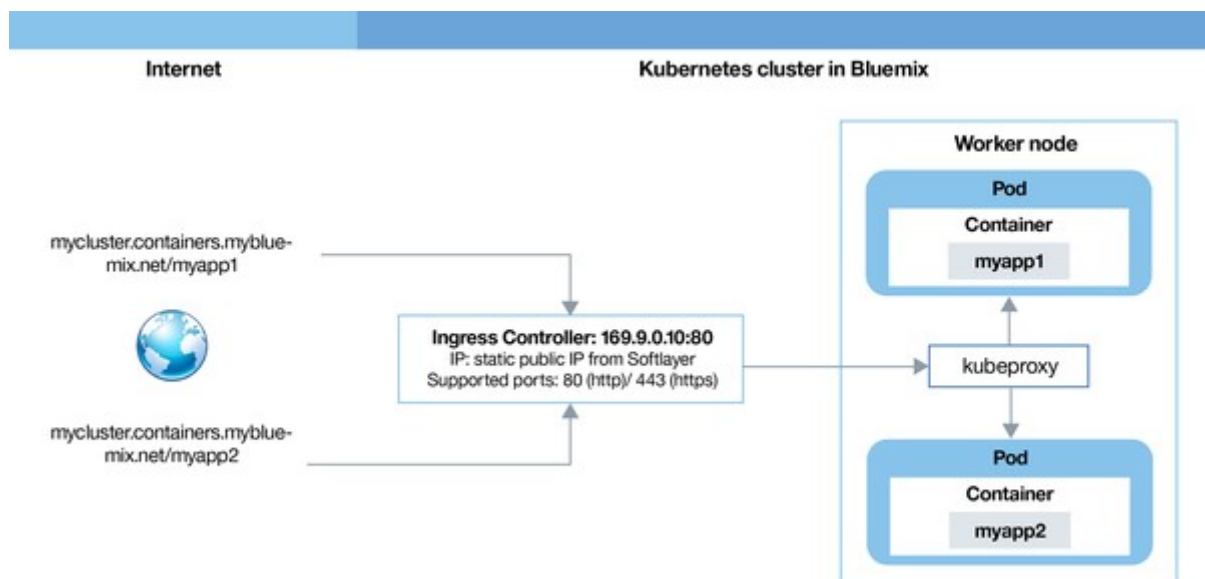
2.4 Namespaces

Kubernetes Namespaces



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 tôi đặt service này là "web" lúc chạy production, còn lúc dev thì tôi chạy nó ở đâu? không nhẽ phải đổi tên service? Namespace giải quyết vấn đề này. Mặc định các dịch vụ sẽ sử dụng namespace "default", nhưng ta muốn tạo namespace nào thì tùy ý. K8s sử dụng 1 namespace riêng : kube-system

2.5 Ingress rules



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

2.6 Network policies

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

2.7 Network

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

2.8 ConfigMaps and Secrets

Một software ít khi chạy luôn mà không cần config. ConfigMap là giải pháp để nhét 1 file config / đặt các ENVIRONMENT var hay set các argument khi gọi câu lệnh. ConfigMap là một cục config, 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 "tôi" mới hardcode mật khẩu vào code. Vậy nên K8s có "secret", để lưu trữ các mật khẩu, token, ... hay những gì cần giữ bí mật.

2.9 Controllers

Các "khái niệm" khác nhau cho các loại dịch vụ khác nhau.

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 thứ tự nhất định.

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.

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 ...).

2.10 Helm - K8s package manager

Trên Ubuntu, ta dùng APT để cài package, thì trên K8s, Helm dùng để cài các "chart", muốn chạy một hệ thống CI ? Install ngay bằng câu lệnh của helm. helm install something

```
$ helm install stable/concourse # cài concourse CI
```

2.11 Dashboard

Dashboard cho phép xem tổng quan về cluster k8s đang dùng, nó được cài vào k8s như một add-on <https://github.com/kubernetes/dashboard> Nói chung là để xem thôi, còn tương tác gì thì cứ câu lệnh kubectl mà chọn.

2.12 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. Hãy tưởng tượng ta có hệ thống monitoring tự động cho mọi pod được tạo - mà không cần làm gì :3 Xem thêm tại <https://kubernetes.io/docs/concepts/cluster-administration/resource-usage-monitoring/>

2.13 Create

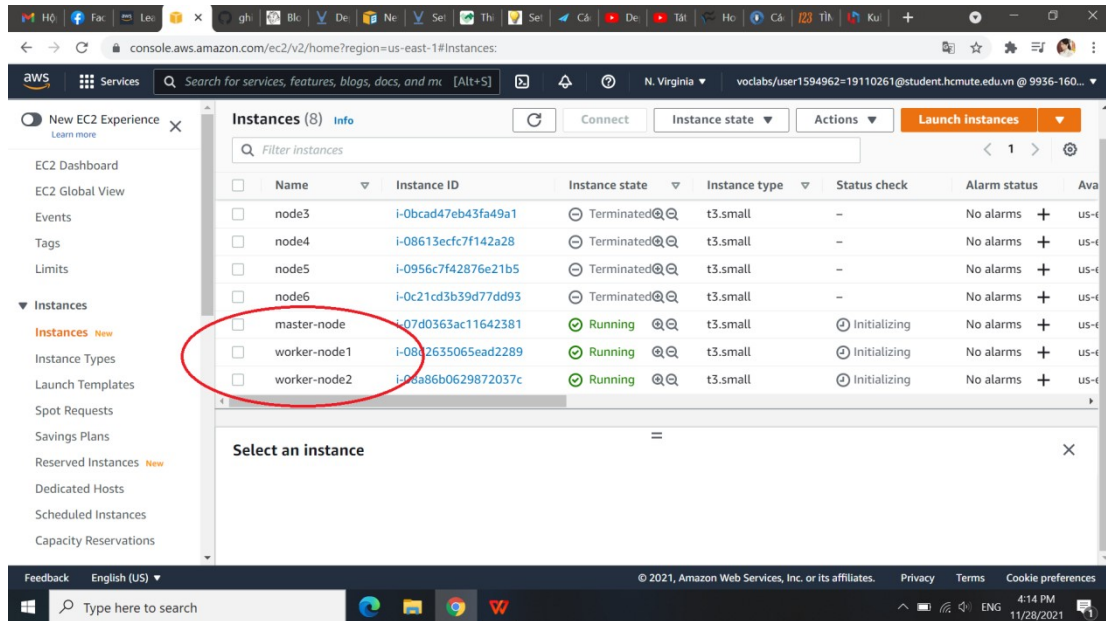
Mọi file cấu hình cho pod/svc/cm/ ... đều là file ở định dạng YAML. Chỉ cần chạy `kubectl create -f filename #` hoặc `.` để cài hết các file trong thư mục hiện tại để deploy chúng.

PHẦN 2: VIẾT VÀ CHẠY ỨNG DỤNG DEMO

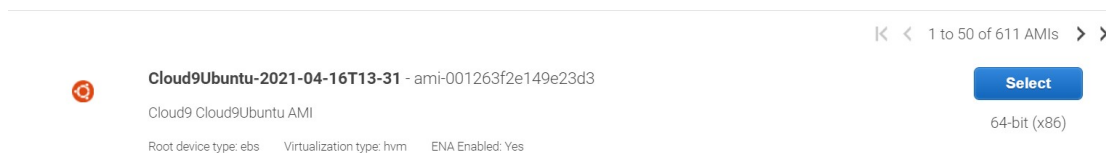
1. Tạo cụm Kubernetes

Chuẩn bị

Nhóm sử dụng dịch vụ Amazon EC2 để tiến hành tạo 3 máy ảo giống nhau



Nền tảng: Cloud9Ubuntu



Loại máy EC2: t3.small

CPU: 02 vCPU

RAM: 02 GB

Khi dùng lệnh “ kubectl join ” nút công nhân không thể tham gia vào nút chính.

Nguyên nhân gốc rễ là cổng 6443, cổng này đã bị chặn trong nhóm bảo mật của AWS.

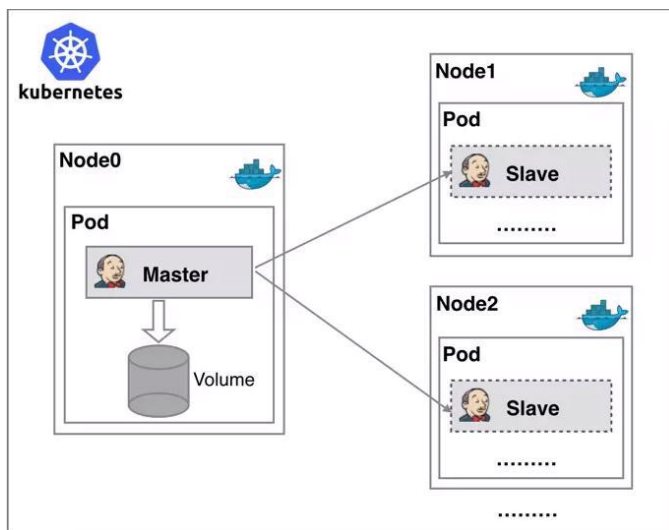
```
root@node2:~# sudo kubectl join 172.31.0.75:6443 --token 93s1kz.dwkqfsxp1qsmb166 \
--discovery-token-ca-cert-hash sha256:f9b41675342635ac6e613a51860ba2c57b957149fc92dada1a49
43f71130fc33
[preflight] Running pre-flight checks
error execution phase preflight: couldn't validate the identity of the API Server: Get "https://172.
31.0.75:6443/api/v1/namespaces/kube-public/configmaps/cluster-info?timeout=10s": net/http: request c
anceled while waiting for connection (Client.Timeout exceeded while awaiting headers)
To see the stack trace of this error execute with --v=5 or higher
root@node2:~#
```

Vì thế em vào Security Groups thêm 1 rule với type: All TCP (giải quyết lỗi)

Inbound rules Info						
Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-06dff15130dcf441b	All TCP	TCP	0 - 65535	Custom <input type="text" value="0.0.0.0/0"/>		Delete
sgr-009c4a1f752b19e71	SSH	TCP	22	Custom <input type="text" value="0.0.0.0/0"/>		Delete

[Add rule](#)

1.1 Mô hình



- Máy chủ 0 = node0 có địa chỉ IP Private là 172.31.1.252
- Máy chủ 1 = node1 có địa chỉ IP Private là 172.31.7.205
- Máy chủ 2 = node2 có địa chỉ IP Private là 172.31.8.111
- Đặt tên cho các máy chủ với lệnh:

```
sudo hostnamectl set-hostname "master"
```

```
sudo hostnamectl set-hostname "node1"
```

```
sudo hostnamectl set-hostname "node2"
```

Trên cả 3 node:

Cài đặt docker

Nhận khóa gpg Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

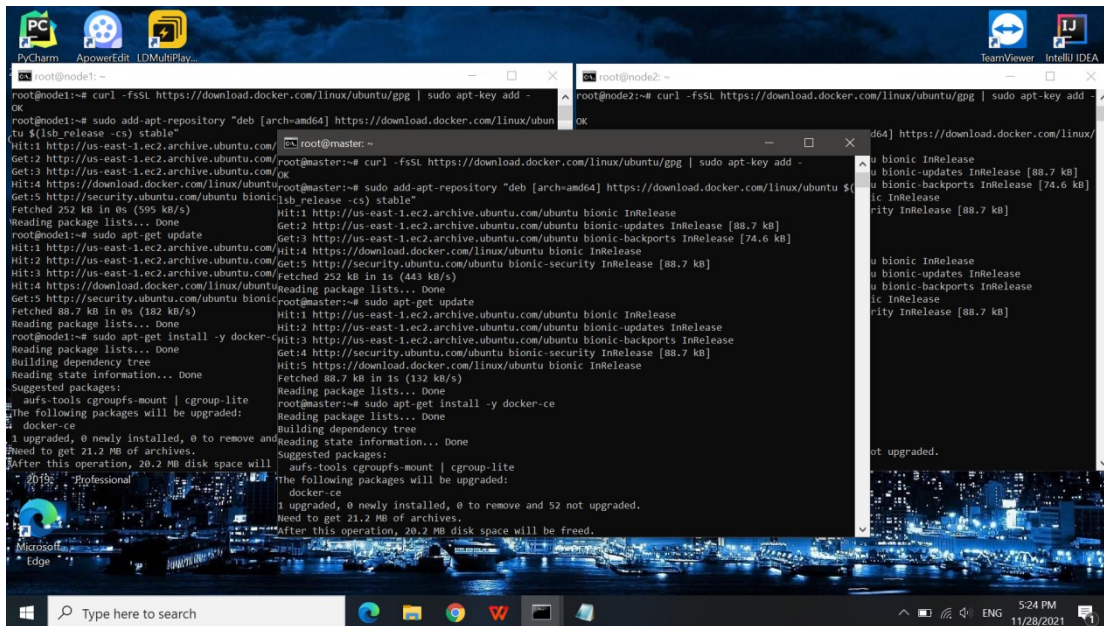
Thêm kho lưu trữ Docker

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

```
sudo apt-get update ( cập nhật các gói )
```

Install Docker

`sudo apt-get install -y docker-ce`



Cài đặt kubelet, kubeadm, kubectl

Nhận khóa gpg Kubernetes

`curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -`

Thêm kho lưu trữ Kubernetes

`cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list`

`deb https://apt.kubernetes.io/ kubernetes-xenial main`

`EOF`

`sudo apt-get update (cập nhật các gói)`

Install kubelet, kubeadm, kubectl

`sudo apt-get install -y kubelet kubeadm kubectl`

Giữ chúng ở phiên bản hiện tại

`sudo apt-mark hold docker-ce kubelet kubeadm kubectl`

Thêm quy tắc iptables vào sysctl.conf

`echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a /etc/sysctl.conf`

Bật iptables ngay lập tức

`sudo sysctl -p`

Đề tắt hoán đổi tạm thời (khắc phục lỗi kubeadm init)

`sudo swapoff -a`

Gặp lỗi với trình điều khiển cgroup

Trình điều khiển cgroup Kubernetes đã được đặt thành hệ thống nhưng docker được đặt thành systemd. Vì vậy, em đã tạo /etc/docker/daemon.json và thêm vào bên dưới:

(khắc phục lỗi **kubeadm init**)

```
sudo touch /etc/docker/daemon.json
```

```
sudo vim /etc/docker/daemon.json
```

```
{  
  "exec-opts": ["native.cgroupdriver=systemd"]  
}
```

```
esc --> :wq
```

```
sudo systemctl daemon-reload
```

```
sudo systemctl restart docker
```

```
sudo systemctl restart kubelet
```

1.2 Tạo Cluster

Cluster sẽ bao gồm các tài nguyên vật lý sau:

- Master node là nút kiểm soát và quản lý một tập hợp các worker node (workloads runtime) và giống như một cluster trong Kubernetes. Nó cũng nắm giữ kế hoạch tài nguyên của node để xác định hành động thích hợp cho event được kích hoạt. Nó chạy etcd, một kho lưu trữ key-value phân tán mã nguồn mở được sử dụng để lưu giữ và quản lý dữ liệu cluster giữa các thành phần lên lịch workload cho các worker node
- 2 worker node là các nút tiếp tục công việc được giao ngay cả khi master node ngừng hoạt động sau khi lập lịch hoàn tất. Các worker node là các máy chủ nơi workload của bạn (tức là các ứng dụng và dịch vụ được tích hợp sẵn) sẽ chạy. Bạn cũng có thể tăng công suất của cluster bằng cách thêm các worker.
- Sau khi tạo thành công, chúng ta sẽ có một cluster đầy đủ chức năng sẵn sàng để chạy workload (tức là các application và service được chứa trong container) nếu các máy chủ trong cluster có đủ tài nguyên CPU và RAM để các ứng dụng của bạn chạy. Sau khi bạn đã thiết lập thành công cluster, bạn có thể chạy hầu hết mọi ứng dụng UNIX truyền thống. Nó có thể được chứa trong cluster của bạn, bao gồm các ứng dụng web, cơ sở dữ liệu, daemon và các công cụ command-line.
- Bản thân cluster sẽ tiêu thụ khoảng 300-500MB bộ nhớ và 10% CPU trên mỗi node.

Trên Master node

Khởi tạo cụm

```
sudo kubeadm init --apiserver-advertise-address=172.31.1.252 --pod-network-cidr=10.244.0.0/16
```

Thiết lập kubeconfig cục bộ

```
mkdir -p $HOME/.kube
```

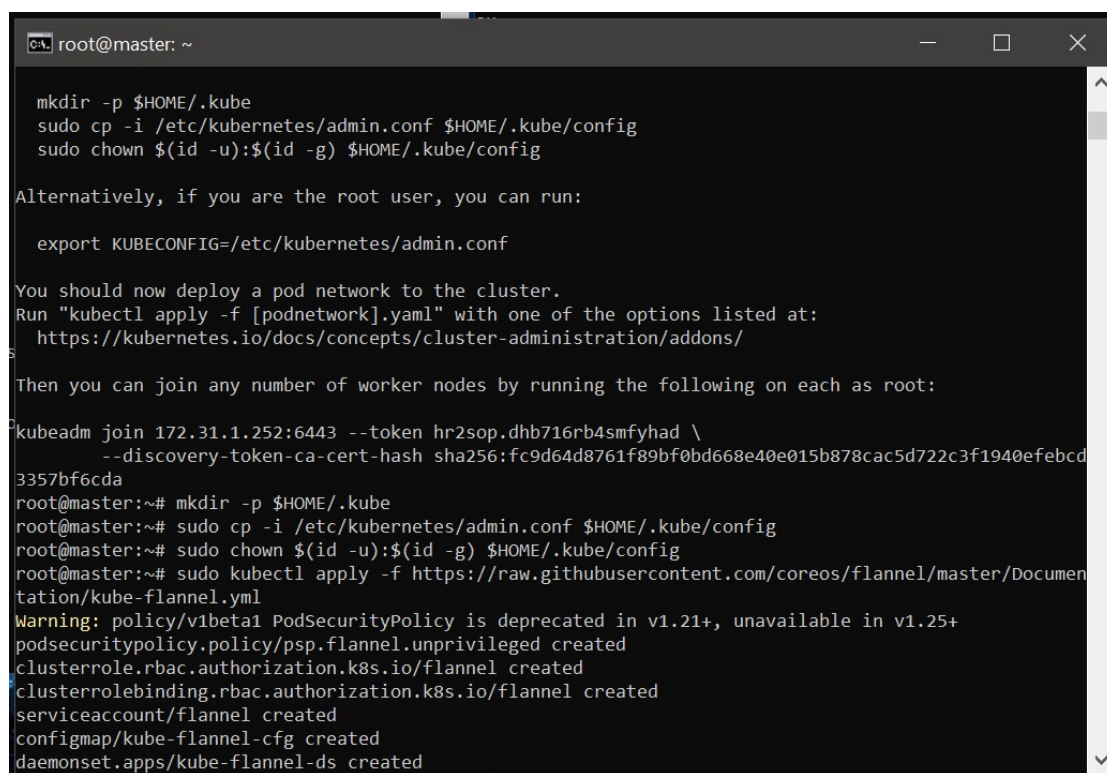
```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Áp dụng lớp phủ mạng CNI Flannel

```
sudo kubectl apply -f
```

<https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>



```
root@master: ~  
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 172.31.1.252:6443 --token hr2sop.dhb716rb4smfyhad \ --discovery-token-ca-cert-hash sha256:fc9d64d8761f89bf0bd668e40e015b878cac5d722c3f1940efebcd3357bf6cda  
root@master:~# mkdir -p $HOME/.kube  
root@master:~# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
root@master:~# sudo chown $(id -u):$(id -g) $HOME/.kube/config  
root@master:~# sudo kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml  
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+  
podsecuritypolicy.policy/psp.flannel.unprivileged created  
clusterrole.rbac.authorization.k8s.io/flannel created  
clusterrolebinding.rbac.authorization.k8s.io/flannel created  
serviceaccount/flannel created  
configmap/kube-flannel-cfg created  
daemonset.apps/kube-flannel-ds created
```

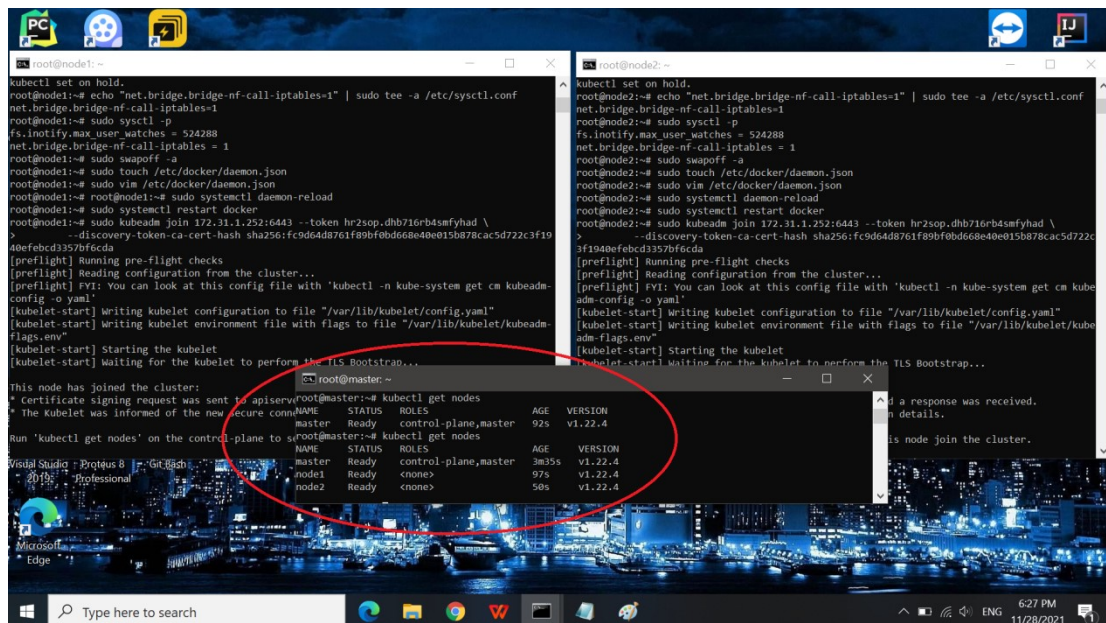
Trên Node1 và Node 2

Các nút công nhân tham gia vào cụm

```
sudo kubeadm join 172.31.1.252:6443 --token hr2sop.dhb716rb4smfyhad \ --discovery-token-ca-cert-hash sha256:fc9d64d8761f89bf0bd668e40e015b878cac5d722c3f1940efebcd3357bf6cda
```

Kiểm tra các node trong Cluster được tạo

```
kubectl get nodes
```



Hoàn thành thiết lập nút chính Kubernetes và nút công nhân trên EC2 của đám mây AWS.

2. Triển khai web NGINX trên Clusters vừa tạo

Trên node master

Tạo triển khai nginx = cách sử dụng NGINX image

`sudo kubectl create deployment nginx --image=nginx`

Tạo 1 dịch vụ với type = NodePort

`sudo kubectl create service nodeport nginx --tcp=80:80`

Xem trạng thái triển khai

`kubectl get deployments`

Xem mô tả triển khai nginx

`kubectl describe deployment nginx`


```
root@master: ~  
root@master:~# sudo kubectl create deployment nginx --image=nginx  
deployment.apps/nginx created  
root@master:~# sudo kubectl create service nodeport nginx --tcp=80:80  
service/nginx created  
root@master:~# kubectl get deployments  
NAME      READY   UP-TO-DATE   AVAILABLE   AGE  
nginx     1/1     1            1           42s  
root@master:~# kubectl describe deployment nginx  
Name:      nginx  
Namespace: default  
CreationTimestamp: Sun, 28 Nov 2021 10:57:42 +0000  
Labels:    app=nginx  
Annotations: deployment.kubernetes.io/revision: 1  
Selector:  app=nginx  
Replicas:  1 desired | 1 updated | 1 total | 1 available | 0 unavailable  
StrategyType: RollingUpdate  
MinReadySeconds: 0  
RollingUpdateStrategy: 25% max unavailable, 25% max surge  
Pod Template:  
  Labels:  app=nginx  
  Containers:  
    nginx:  
      Image:      nginx  
      Port:       <none>  
      Host Port:  <none>  
      Environment: <none>  
      Mounts:     <none>  
      Volumes:    <none>
```

Xem các dịch vụ đang có

kubectl get svc

Kiểm tra số pod

kubectl get pods

Scale lên 10 pod

kubectl scale --replicas=10 deployments/nginx

Các pods của nginx-deployment được Cluster chia đều qua 2 worker node.

Câu hỏi đặt ra, điều gì 1 server worker node bị lỗi or chết?

Em sẽ tiến hành thực hành để thấy rõ vấn đề này.

Thử xóa 1 pods

kubectl delete pod nginx-6799fc88d8-gb8zd

Kiểm tra lại số pod

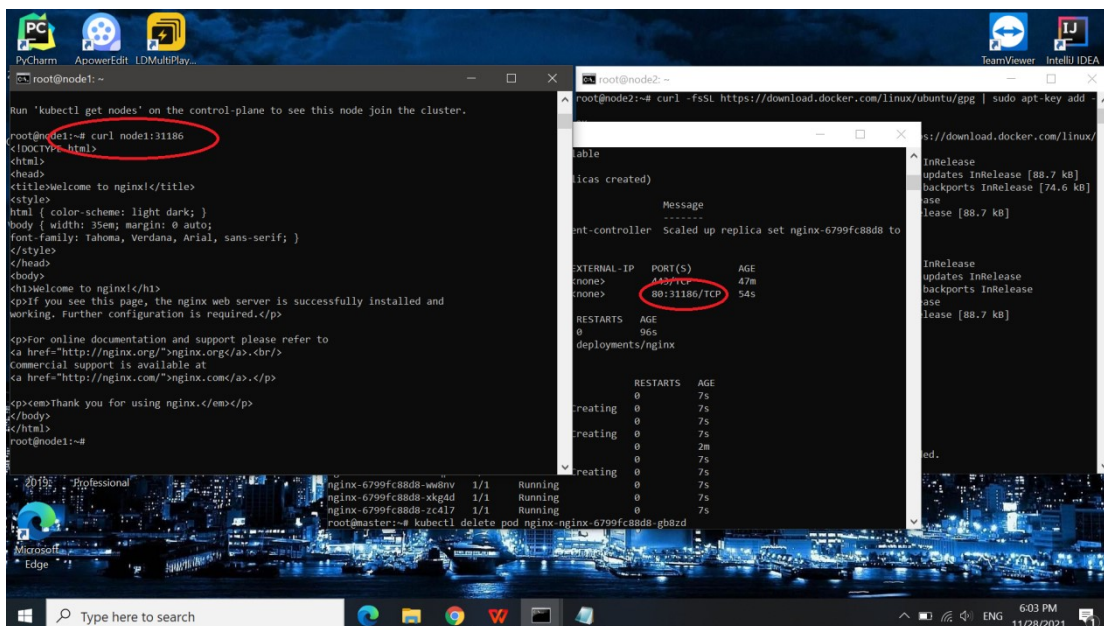
kubectl get pods

Thì thấy 1 pod nginx-6799fc88d8-gb8zd đã bị xóa và 1 pods mới nginx-6799fc88d8-qt529 được tạo ra sau 7s thay thế cho pod cũ nginx-6799fc88d8-gb8zd

```
root@master: ~  
root@master:~# kubectl get svc  
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE  
kubernetes ClusterIP   10.96.0.1        <none>           443/TCP         47m  
nginx      NodePort    10.100.118.152   <none>           80:31186/TCP   54s  
root@master:~# kubectl get pods  
NAME      READY   STATUS    RESTARTS   AGE  
nginx-6799fc88d8-stmsv 1/1     Running   0          96s  
root@master:~# kubectl scale --replicas=10 deployments/nginx  
deployment.apps/nginx scaled  
root@master:~# kubectl get pods  
NAME      READY   STATUS    RESTARTS   AGE  
nginx-6799fc88d8-gb8zd 1/1     Running   0          7s  
nginx-6799fc88d8-jn55q 0/1     ContainerCreating 0          7s  
nginx-6799fc88d8-kt629 1/1     Running   0          7s  
nginx-6799fc88d8-skqw2 0/1     ContainerCreating 0          7s  
nginx-6799fc88d8-stmsv 1/1     Running   0          2m  
nginx-6799fc88d8-t46xr 1/1     Running   0          7s  
nginx-6799fc88d8-vtqp8 0/1     ContainerCreating 0          7s  
nginx-6799fc88d8-ww8nv 1/1     Running   0          7s  
nginx-6799fc88d8-xkg4d 1/1     Running   0          7s  
nginx-6799fc88d8-zc4l7 1/1     Running   0          7s  
root@master:~# kubectl delete pod nginx-nginx-6799fc88d8-gb8zd  
Error from server (NotFound): pods "nginx-nginx-6799fc88d8-gb8zd" not found  
root@master:~# kubectl delete pod nginx-6799fc88d8-gb8zd  
pod "nginx-6799fc88d8-gb8zd" deleted  
root@master:~# kubectl get pods  
NAME      READY   STATUS    RESTARTS   AGE  
nginx-6799fc88d8-jn55q 1/1     Running   0          2m33s  
nginx-6799fc88d8-kt629 1/1     Running   0          2m33s  
nginx-6799fc88d8-qt529 1/1     Running   0          7s  
nginx-6799fc88d8-skqw2 1/1     Running   0          2m33s  
nginx-6799fc88d8-stmsv 1/1     Running   0          4m26s  
nginx-6799fc88d8-t46xr 1/1     Running   0          2m33s  
nginx-6799fc88d8-vtqp8 1/1     Running   0          2m33s  
nginx-6799fc88d8-ww8nv 1/1     Running   0          2m33s  
nginx-6799fc88d8-xkg4d 1/1     Running   0          2m33s  
nginx-6799fc88d8-zc4l7 1/1     Running   0          2m33s  
root@master:~#
```

Thực hiện thành công theo đúng như lý thuyết điển hình về Kubernetes Cluster

Truy cập web trên worker node



Mở trên browser

Sau khi em tìm hiểu thì

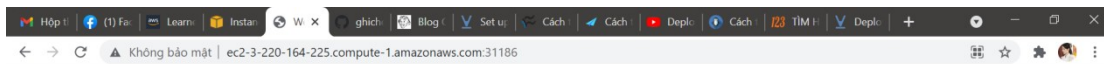
Dùng lệnh: # ip a

Thì thấy không có IP Public nào tồn tại.

Vấn đề: Vì hiện đang chạy triển khai này trên Máy ảo do nhà cung cấp đám mây công cộng cung cấp. Vì vậy, mặc dù không có giao diện cụ thể nào được chỉ định IP công cộng, nhưng nhà cung cấp máy ảo đã cấp một địa chỉ IP bên ngoài Ephemeral.

Địa chỉ IP bên ngoài tạm thời là địa chỉ IP tạm thời vẫn được gắn vào máy ảo cho đến khi phiên bản ảo bị dừng. Khi phiên bản ảo được khởi động lại, một IP bên ngoài mới sẽ được chỉ định. Về cơ bản, đó là một cách đơn giản để các nhà cung cấp dịch vụ tận dụng các IP công cộng nhàn rỗi.

Thách thức ở đây, ngoài thực tế là IP công cộng không phải là tĩnh, IP công cộng Ephemeral chỉ đơn giản là một phần mở rộng (hoặc proxy) của IP Riêng và vì lý do đó, dịch vụ sẽ chỉ được truy cập trên cổng **31186**. Điều đó có nghĩa là dịch vụ sẽ được truy cập trên URL, đó là <http://ec2-3-220-164-225.compute-1.amazonaws.com:31186/>, kiểm tra trình duyệt của mình, em đã thấy trang chào mừng.



Em đã triển khai thành công NGINX trên cụm Kubernetes 3 nút của chúng em.
<http://ec2-3-220-164-225.compute-1.amazonaws.com:31186/>

Vì mỗi lần reset thì sẽ thay đổi Public IPv4 DNS nên sẽ k còn dùng được nữa.

Public IPv4 DNS

 ec2-3-220-164-225.compute-
1.amazonaws.com | [open address](#) 

Answer private resource DNS name

IPv4 (A)

PHẦN 3: KẾT LUẬN

1. Nội dung đã thực hiện được

- Tìm hiểu khá đầy đủ về Kubernetes
- Sử dụng thành công lệnh kubectl để khởi tạo cluster trên máy ảo EC2
- Chạy demo thành công 1 ứng dụng web (mã nguồn mở) bằng Kubernetes và nêu các tính chất đặc trưng, điển hình của Kubernetes.

2. Vấn đề còn tồn đọng

- Hiện tại không thể deploy web tự tạo mà phải dùng web mã nguồn mở có sẵn Nginx trên DockerHub để tiện deploy và mô tả Kubernetes.

3. Hướng phát triển đề tài

- Mở rộng mô hình cụm Kubernetes ứng dụng với các project lớn.

PHẦN 4 : TÀI LIỆU THAM KHẢO

1. Lý thuyết

<https://kubernetes.io/vi/docs/concepts/overview/what-is-kubernetes/>
<https://viblo.asia/p/phan-1-gioi-thieu-ve-kubernetes-924lJO6m5PM>
<https://viblo.asia/p/phan-2-kien-truc-cua-kubernetes-RQqKLnr6l7z>
https://hocdevops.com/kubernetes/kien-truc-cua-kubernetes/#Kien_truc_Kubernetes_Cac_thanh_phan_Kubernetes
<https://www.thegioimaychu.vn/blog/ao-hoa/kubernetes-gioi-thieu-kien-truc-va-cach-thuc-hoat-dong-p5642/>
<https://topdev.vn/blog/kubernetes-la-gi/>
<https://viblo.asia/p/ly-thuyet-tim-hieu-co-ban-ve-kubernetes-cac-thanh-phan-chinh-trong-mot-kubernetes-cluster-va-cac-khai-niem-co-ban-kem-video-WAyK8D7EKxX>
<https://blog.itnavi.com.vn/kubernetes-la-gi/>
<https://www.youtube.com/watch?v=xRj22tG4J9w&t=2s>
<https://www.youtube.com/watch?v=yOBeQNGX278>
https://www.youtube.com/watch?v=_vgUxbsRoLY&t=2190s

2. Thực hành

https://www.suse.com/c/rancher_blog/kubernetes-deployment-how-to-run-a-containerized-workload-on-a-cluster/?fbclid=IwAR1t10vwQMExW0p9FkYMi9BrGRvQ6nRcI3lflGSu9XYJ1eNmVufNOYkDv_I
https://viblo.asia/p/deploy-ung-dung-dau-tien-cua-ban-len-kubernetes-naQZRLz05vx?ref=morioh.com&utm_source=morioh.com&fbclid=IwAR1wFRL8Np4mtMac7Z404xuzN9WwkXqqPhK2N514Rvqsos4dALJBwecvZxQ
<https://github.com/hocchudong/ghichep-kubernetes/blob/master/notes/kubeadm.md>
<https://helpex.vn/article/thiet-lap-kubernetes-master-va-worker-node-tren-amazon-web-service-aws-608bec4e7b71350b9c1bad36>
<https://www.howtoforge.com/setup-a-kubernetes-cluster-on-aws-ec2-instance-ubuntu-using-kubeadm/>
https://www.suse.com/c/rancher_blog/kubernetes-deployment-how-to-run-a-containerized-workload-on-a-cluster/
<https://docs.bitnami.com/tutorials/deploy-static-content-apache-kubernetes/>

<https://www.bluematador.com/blog/building-and-deploying-to-kubernetes>

<https://www.section.io/engineering-education/deploy-docker-container-to-kubernetes-cluster/>

<https://www.techrepublic.com/article/how-to-deploy-nginx-on-a-kubernetes-cluster/>

<https://www.middlewareinventory.com/blog/deploy-docker-image-to-kubernetes/>