**ALL ABOUT KUBERNETES**

**I.Node :**

- Node là một máy tính có thể là hardware machine hoặc là một virtual machine.

- Các node này sẽ được tổ chức nằm trong các cluster, có các service cần thiết để có thể chạy các Pod, chịu sự quản lý của các cluster đó và được sử dụng tuỳ theo nhu cầu. Bất kỳ một node nào cũng có thể được thay thế bởi những node khác trong cùng một cluster.

- Để kiểm tra các node hiện tại có trong cluster của Kubernetes, các bạn hãy nhập command sau: **kubectl get nodes**

**II.Cluster :**

* cluster sẽ là một hệ thống bao gồm nhiều node khác nhau. Cluster sẽ đảm nhận việc quản lý những node này.

**III.POD :**

* Pod là khái niệm dùng để chỉ một hoặc nhiều container đang chạy trong Kubernetes. Nếu nhiều container thì chúng sẽ sử dụng cùng resource và network để dễ dàng communication với nhau.
* Một pod có thể được replicate thành nhiều bản trong Kubernetes.
* Chúng ta có 2 cách để tạo mới một Pod trong Kubernetes đó là chạy container từ Image sử dụng câu lệnh “run” hoặc sử dụng câu lệnh “create” với một tập tin cấu hình YAML định nghĩa thông tin của Pod. Các bạn có thể định nghĩa thông tin của một Pod sử dụng tập tin có nội dung cơ bản như sau:



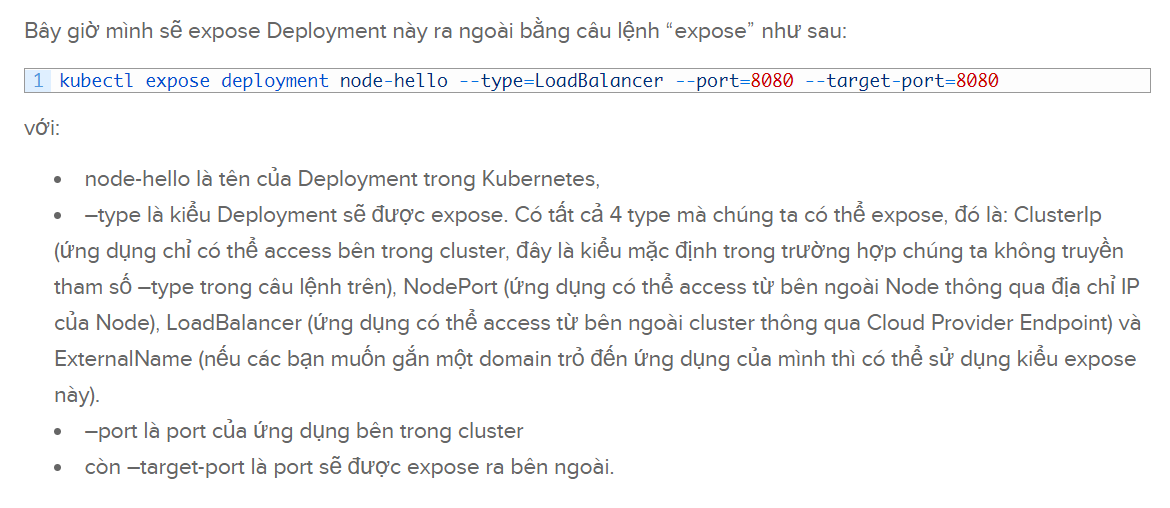
**IV.Deployment**

* Các pod sẽ không được chạy trực tiếp trong Kubernetes mà nó phải thông qua một thành phần nữa gọi là Deployment. Việc này đảm bảo cho việc chúng ta có thể replacate các pods với số lượng được khai báo trước. Khi một Deployment được thêm vào, nó sẽ khởi tạo một số lượng các pods mà chúng ta cần và quản lý chúng.
* Deployment trong Kubernetes có 3 tham số định nghĩa strategy để chúng ta có thể update một Deployment.
* Thứ nhất là tham số minReadySecond định nghĩa bao lâu thì một Pod vừa mới tạo được xem xét là available. Mặc định thì Kubernetes sẽ coi Pod available khi nó chạy thành công nên giá trị mặc định của tham số này là 0.
* Tham số thứ 2 maxSurged định nghĩa số lượng Pod có thể tăng lên cho Deployment này là bao nhiêu. Giá trị của tham số này có thể là một số bất kỳ hoặc tính theo %. Mặc định giá trị này sẽ là 25% so với số lượng Pod cần thiết để chạy ứng dụng các bạn nhé!
* Tham số thứ 3 liên quan đến việc bao nhiêu Pod sẽ trở nên unavailable khi Deployment đang được cập nhập. Giá trị của tham số này cũng có thể là một số bất kỳ hoặc tính theo %. Và mặc định giá trị này cũng sẽ là 25% các bạn nhé!

**V.Ingress**

- Sau khi ứng dụng của chúng ta được deploy lên các container được quản lý bởi Kubernetes cluster, làm thế nào để chúng ta truy cập được tới nó?

Chúng ta có nhiều cách để thêm một Ingress vào một cluster, ví dụ như sử dụng Ingress Controller hay Load Balancer.

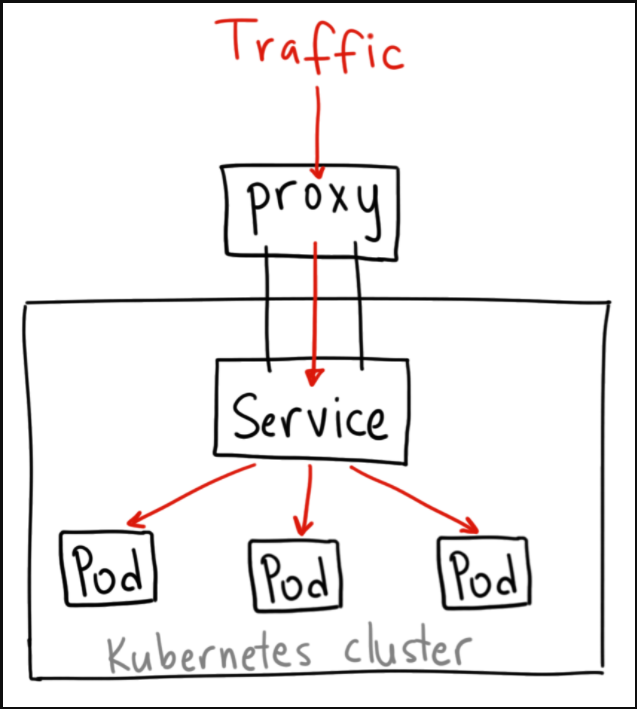




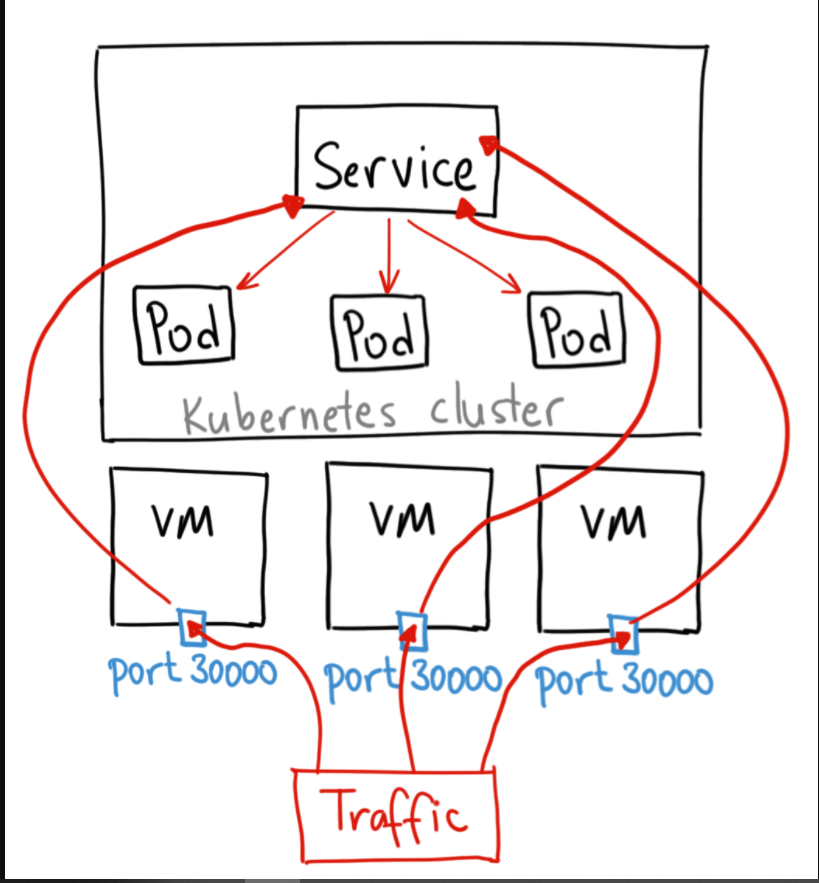
**VI.Network**

-Cluster IP :

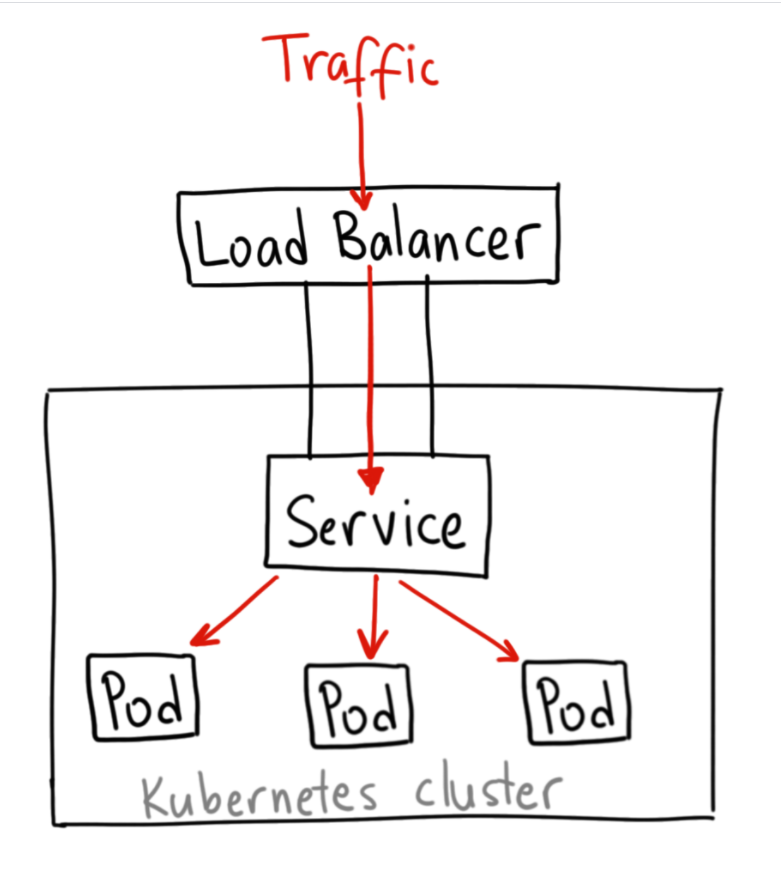
Loại đơn giản nhất, loại mặc định. Sẽ mở quyền truy cập vào một ứng dụng trong một cụm, mà không cần quyền truy cập từ thế giới.



-Nodeport :

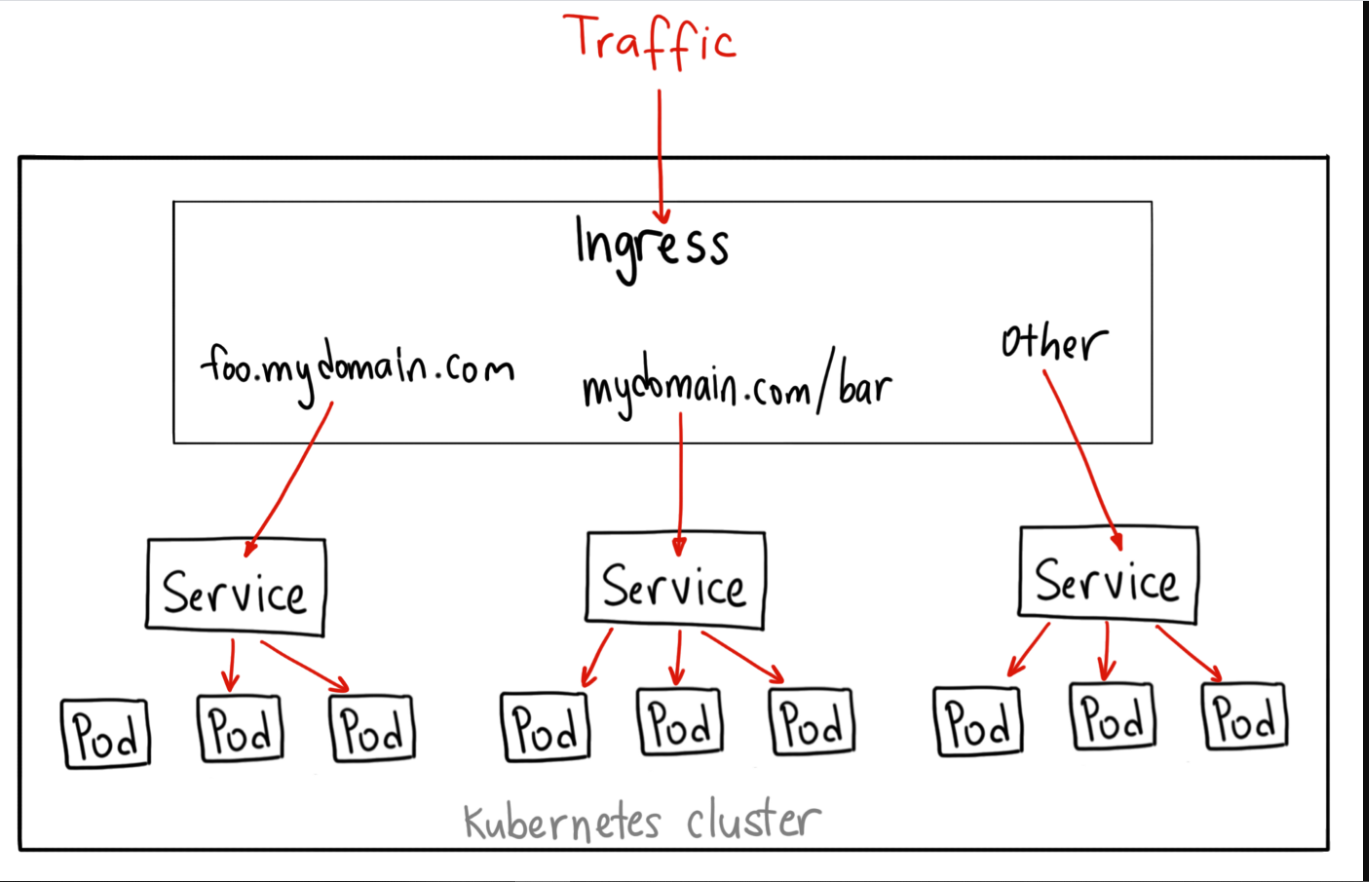


* LoadBalancer



* Ingress

<https://hocweb.vn/kubernetes-service-la-gi-tai-sao-phai-su-dung-kubernetes-phan-2/>



Cơ chế kiểm tra container **Liveness** giúp **Kubernetes** biết là ứng dụng container của bạn đang sống hay chết (hay không hoạt động đúng chức năng).

Vậy chúng ta sẽ cần “**Rediness probe**” là cơ chế giúp [**Kubernetes**](https://cuongquach.com/tag/kubernetes) kiểm tra xem ứng dụng container của bạn đã sẵn sàng tiếp nhận xử lý request chưa

**Các cơ chế kiểm tra health check Liveness/Readiness**

* **HTTP probe**: kubelet sẽ gửi HTTP request đến host (mặc định sẽ là Pod Ip) và url mà bạn chỉ định. Nếu mà dịch vụ web service của bạn phản hồi HTTP Status 200 hay 300 thì tức ứng dụng container của bạn vẫn hoạt động tốt. Ngược lại thì là ứng dụng container của bạn chưa sẵn sàng phục vụ request hoặc đang bị lỗi cần restart lại.
* **TCP probe**: kubelet sẽ thử khởi tạo connection TCP đến port quy định. Nếu khởi tạo thành công thì ứng dụng container của bạn đã sẵn sàng hoặc vẫn còn hoạt động tốt.
* **Command**: sẽ thực hiện chương trình command hoặc script bạn chỉ định. Exit code của command cần trả về là 0 thì là thành công.

**Một số cấu hình lưu ý Liveness/Readiness Probe**

Một số trường cấu hình bạn nên nắm qua ý nghĩa của nó khi sử dụng cấu hình **Liveness**/**Readiness** :

* **initialDelaySeconds**: thời gian (giây) sau khi container khởi động thì sẽ bắt đầu các hoạt động kiểm tra liveness/readiness. Mặc định là 0 giây.
* **periodSeconds**: thời gian giữa các lần thực hiện kiểm tra liveness/readiness. Mặc định là 10 giây.
* **timeoutSeconds**: thời gian timeout của các hoạt động kiểm tra liveness/readiness. Mặc định là 1 giây.
* **successThreshold**: số lần thực hiện kiểm tra liveness/readiness trả kết quả tốt thì được coi là thành công cho việc kiểm tra. Mặc định là 1 .
* **failureThreshold**: số lần thực hiện kiểm tra liveness/readiness trả kết quả xấu thì được coi là thất bại cho việc kiểm tra. Mặc định là 3.

**VII.** **Horizontal pod auto scaling ( HPA )**

HPA viết tắt từ Horizontal Pod Autoscaler - có thể hiểu đây là một bộ điều hành việc scale theo chiều ngang các Pod. Yêu cầu :

1. Metrics Server :

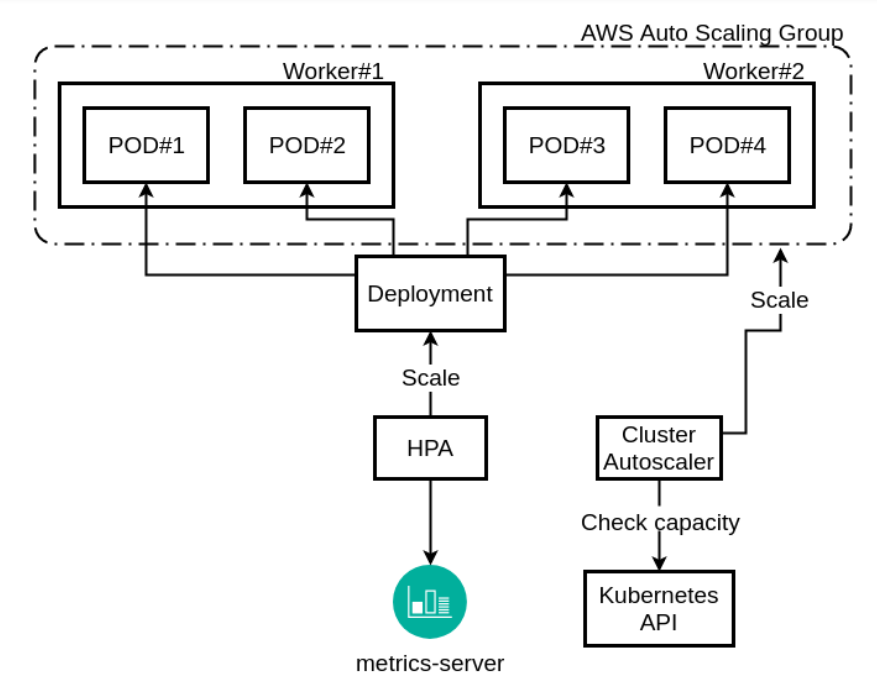
Metrics Server là một server tập hợp lại các metrics (chỉ số đo lường) của các container (các pods) phục vụ cho chu trình autoscaling tích hợp trong K8S

1. Cluster Auto-Scaler :

Cluster Auto-Scaler là một chức năng trong K8S, chịu trách nhiệm tăng / hoặc giảm số lượng của node sao cho phù hợp với số lượng pods vận hành.

1. Thiết lập các thông số Requests/Limits của tài nguyên và xác định trạng thái Liveness/Readiness :

HPA dựa vào lượng % CPU được sử dụng để tính toán ra số pod cần scale, nên ta cầ phải hiểu được cách thức tính toán từ đó xác định được các thông số request/limit phù hợp cho từng pod.



* Metrics server tổng hợp các thông số từ các pods hiện tại
* HPA sẽ check các thông số trên 15s một lần, nếu giá trị rơi vào khoảng thiết lập trong HPA thì nó sẽ tiến hành tăng/ giảm số lượng pods
* Trong trường hợp scale-up, (Một bộ phận tên) Kubernetes scheduler sẽ tạo một pods vào node mà có đủ tài nguyên mà pod yêu cầu
* Nếu không đủ tài nguyên, Cluster auto-scaler sẽ tăng số node lên để đáp ứng với lượng pod mà đang được dự định tạo mới
* Trong trường hợp scale-down, HPA sẽ giảm số lượng pod
* Cluster scaler nếu thấy các node đang "rảnh", và pods của 1 node X nào đó có thể chuyển sang một node khác để tận dụng tài nguyên thì pod sẽ chuyển sang, và node X đó sẽ được xóa (scale-down) đi)

Các phiên bản API cho các đối tượng HorizontalPodAutoscaler . Khi bạn sử dụng Google Cloud Console, các đối tượng HorizontalPodAutoscaler được tạo bằng cách sử dụng API autoscaling / v2beta2. Khi bạn sử dụng kubectl để tạo hoặc xem thông tin về Bộ tự động định tỷ lệ ngang, bạn có thể chỉ định API autoscaling / v1 hoặc API autoscaling / v2beta2.

* apiVersion: **autoscaling / v1** là mặc định và cho phép bạn chỉnh tỷ lệ tự động chỉ dựa trên việc sử dụng CPU. Để tự động phân tỷ lệ dựa trên các số liệu khác, bạn nên sử dụng apiVersion: autoscaling / v2beta2. Ví dụ trong Định cấu hình triển khai sử dụng apiVersion: autoscaling / v1.
* apiVersion: **autoscaling / v2beta2** được khuyến nghị để tạo các đối tượng HorizontalPodAutoscaler mới. Nó cho phép bạn phân tỷ lệ tự động dựa trên nhiều số liệu, bao gồm cả số liệu tùy chỉnh hoặc bên ngoài. Tất cả các ví dụ khác trong chủ đề này sử dụng apiVersion: autoscaling / v2beta2.

**VIII.** **Vertical pod auto scaling ( VPA )**

VPA giúp bạn không phải suy nghĩ về những giá trị nào cần chỉ định cho các yêu cầu và giới hạn CPU của vùng chứa cũng như các yêu cầu và giới hạn bộ nhớ. Bộ định mức tự động có thể đề xuất các giá trị cho các yêu cầu và giới hạn của CPU và bộ nhớ hoặc nó có thể tự động cập nhật các giá trị. Tính năng tự động thay đổi tỷ lệ dọc Pod cung cấp những lợi ích sau:

* Các nút cụm được sử dụng hiệu quả vì Nhóm sử dụng chính xác những gì họ cần.
* Các nhóm được lập lịch trên các nút có sẵn các tài nguyên thích hợp.
* Bạn không phải chạy các tác vụ đođiểm chuẩn tốn thời gian để xác định các giá trị chính xác cho các yêu cầu CPU và bộ nhớ.
* Thời gian bảo trì được giảm xuống, vì trình tự động phân tích có thể điều chỉnh các yêu cầu CPU và bộ nhớ theo thời gian mà không cần bạn thực hiện bất kỳ hành động nào.

Chúng ta có thể sử dụng vpa để xem recommendation về CPU hay memory so với ban đầu chúng ta mặc định . Khi đó thì update policy được chuyển ở chế độ off trong file yaml của vpa . Nếu chúng ta muốn vpa tự update cho chúng ta theo recommendation của nó thì chúng ta chuyển nó về 0 ( nó chỉ áp dụng cho 2 replica trở lên và nó chỉ tác động ở pod chứ không liên quan đến deploy )

<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>