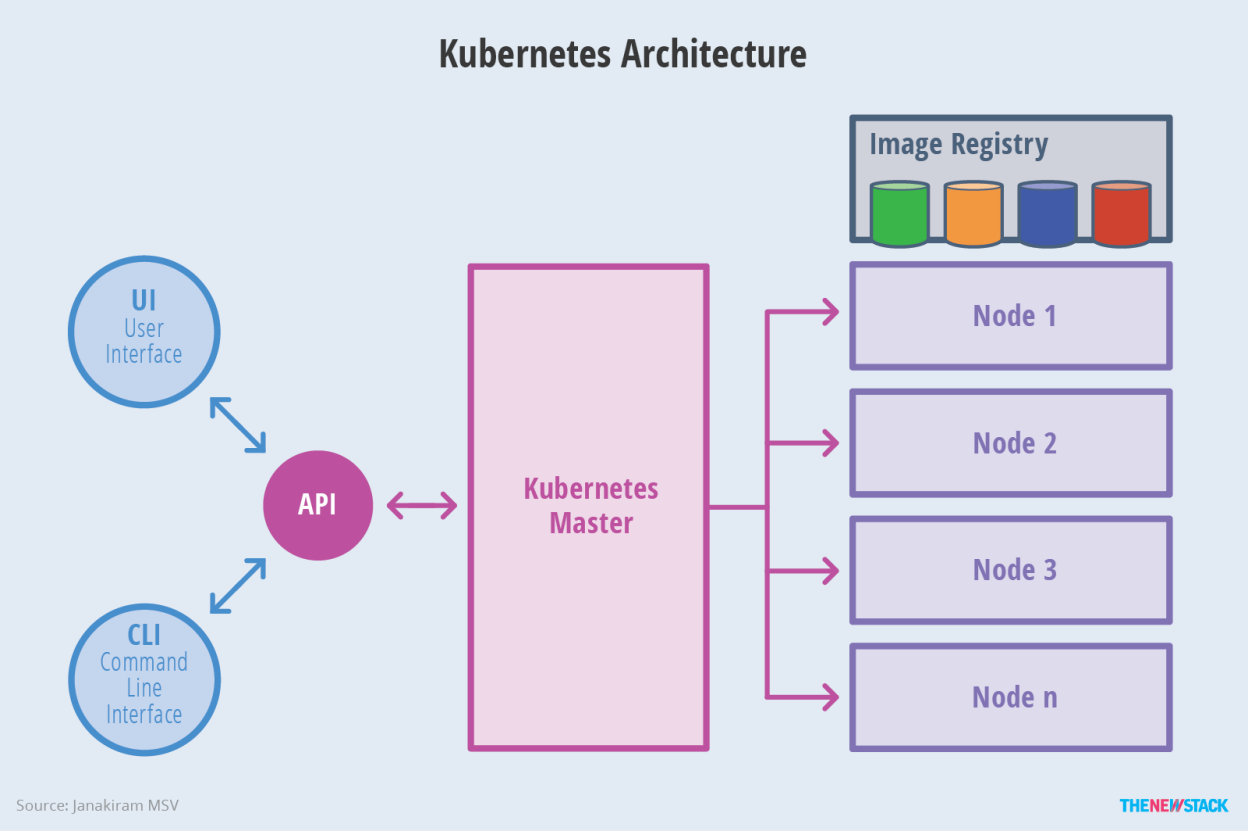
# Tìm hiểu Kubernetes và triển khai thử nghiệm

## 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**.

## 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.

## 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:

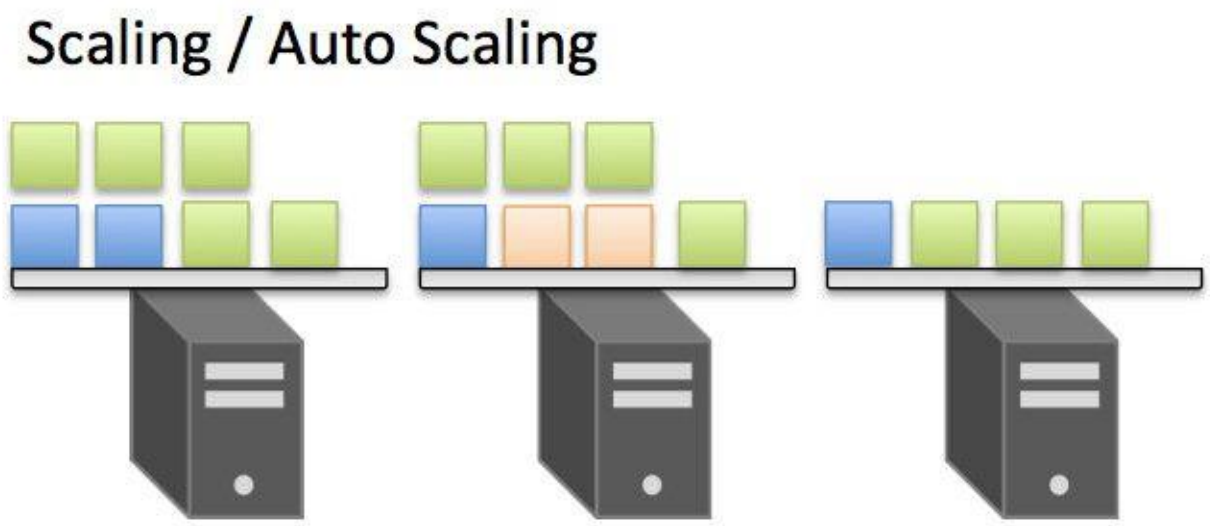
- Việc quản lý hàng loạt docker host
- Container Scheduling
- Rolling update
- Scaling/Auto Scaling
- Monitor vòng đời và tình trạng sống chết của container.
- Self-hearing trong trường hợp có lỗi xãy ra. (Có khả năng phát hiện và tự correct lỗi)
- Service discovery
- Load balancing
- Quản lý data, work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

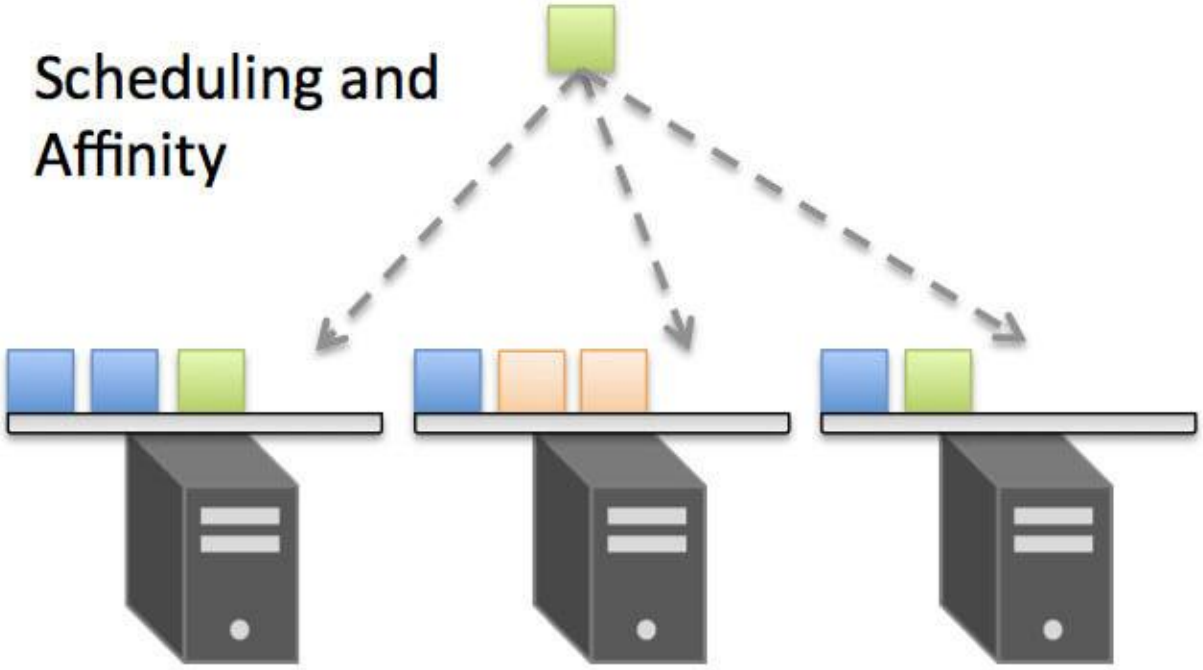
K8s quản lý thực thi các container sử dụng YAML để viết các Manifest.

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx-server
spec:
  replicas: 1
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: nginx-server
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: "app"
                operator: In
                values:
                  - develop
      containers:
      - name: nginx
        image: "nginx:latest"
        imagePullPolicy: "Always"
        ports:
          - containerPort: 80
        volumeMounts:
        - mountPath: /etc/nginx
          name: nginx-conf
        - mountPath: /var/log/nginx
          name: nginx-log
```

Sau khái niệm kubernetes là gì chúng ta hãy đến với chức năng của nó. Kubernetes quản lý các docker host và cấu trúc container cluster. Ngoài ra, khi thực thi các container trên K8s, bằng cách thực hiện replicas (tạo ra nhiều container giống nhau) làm cho hệ thống có sức chịu lỗi cao và tự động thực hiện load balancing. Thông qua cơ chế load balancing, chúng ta có thể tăng giảm số lượng container replica (auto scaling).

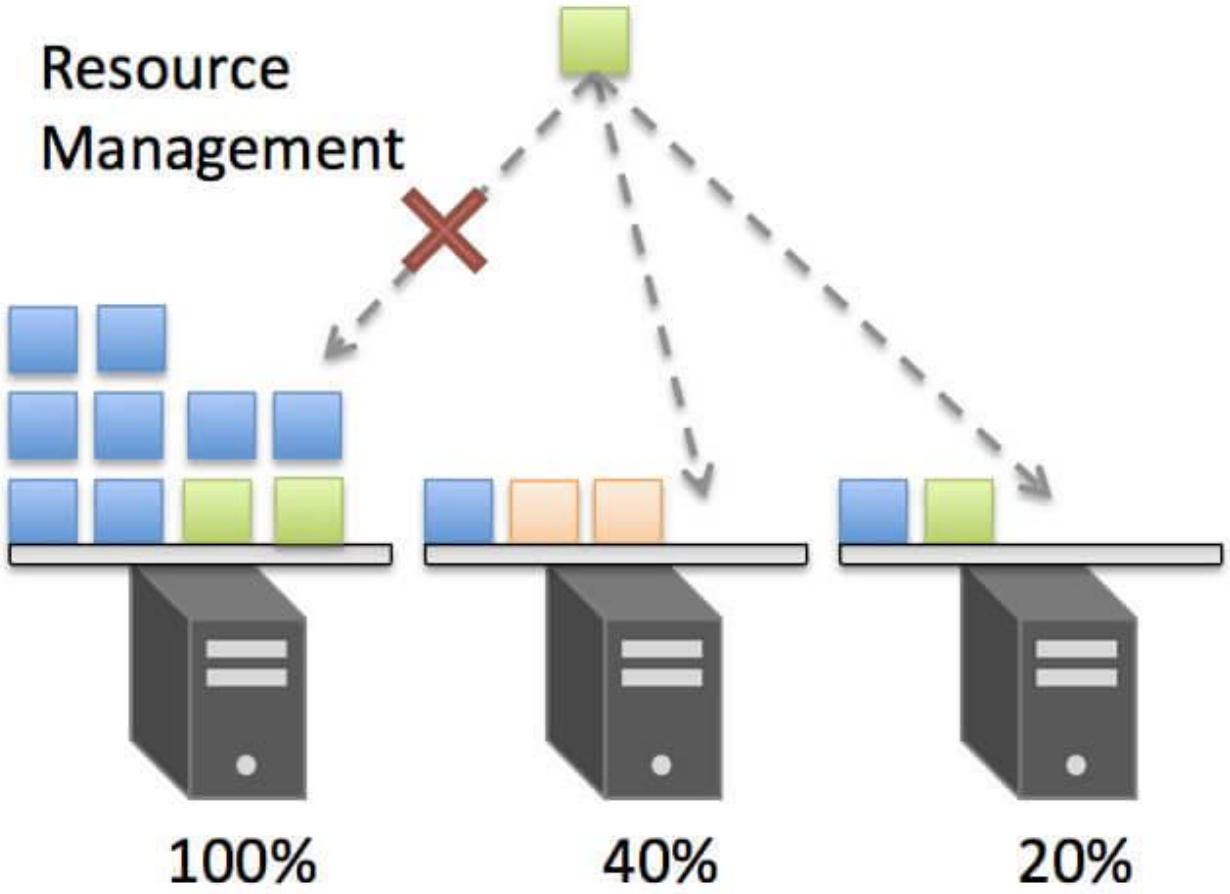


Khi thực hiện phân chia container vào các Node (docker host), dựa trên các loại docker host kiểu như “Disk SSD” hay “số lượng clock của CPU cao”… Hoặc dựa trên loại Workload kiểu như “Disk I/O quá nhiều”, “Băng thông đến một container chỉ định quá nhiều” … **K8s** sẽ ý thức được việc affinity hay anti-affinity và thực hiện Scheduling một cách hợp lý cho chúng ta.



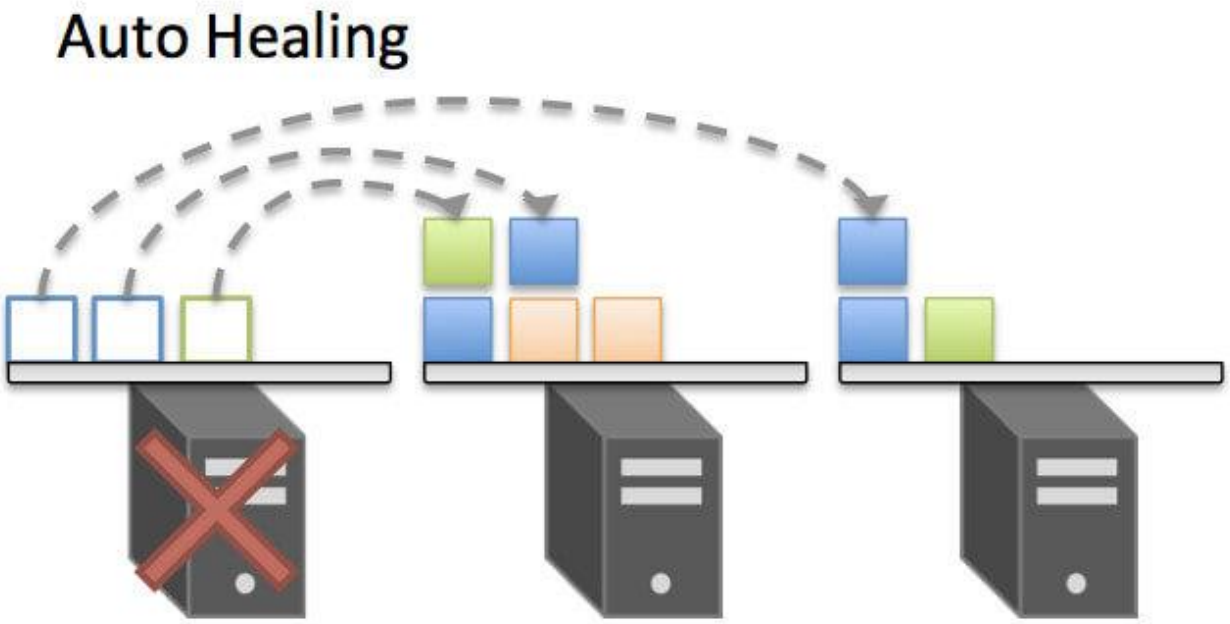

Trong trường hợp không được chỉ định host cụ thể, K8s sẽ thực hiện scheduling tuỳ thuộc vào tình trạng CPU, memmory của docker host có trống hay không. Vì vậy, chúng ta không cần quan tâm đến việc quản lý bố trí container vào các docker host như thế nào.

Hơn nữa, trường hợp resource không đủ, thì việc auto scheduling của K8s cluster cũng sẽ được thực hiện tự động.




Được xây dựng theo quan điểm tính chịu lỗi cao, **K8s** thực hiện monitor các container theo tiêu chuẩn. Trong trường hợp bất ngờ nào đó, khi một container process bị dừng, **K8s** sẽ thực hiện **Self-hearing** bằng cách scheduling một container nữa.

Thêm nữa, ngoài việc monitor hệ thống, k8s còn có khả năng thiết lập health check bằng HTTP/TCP script.

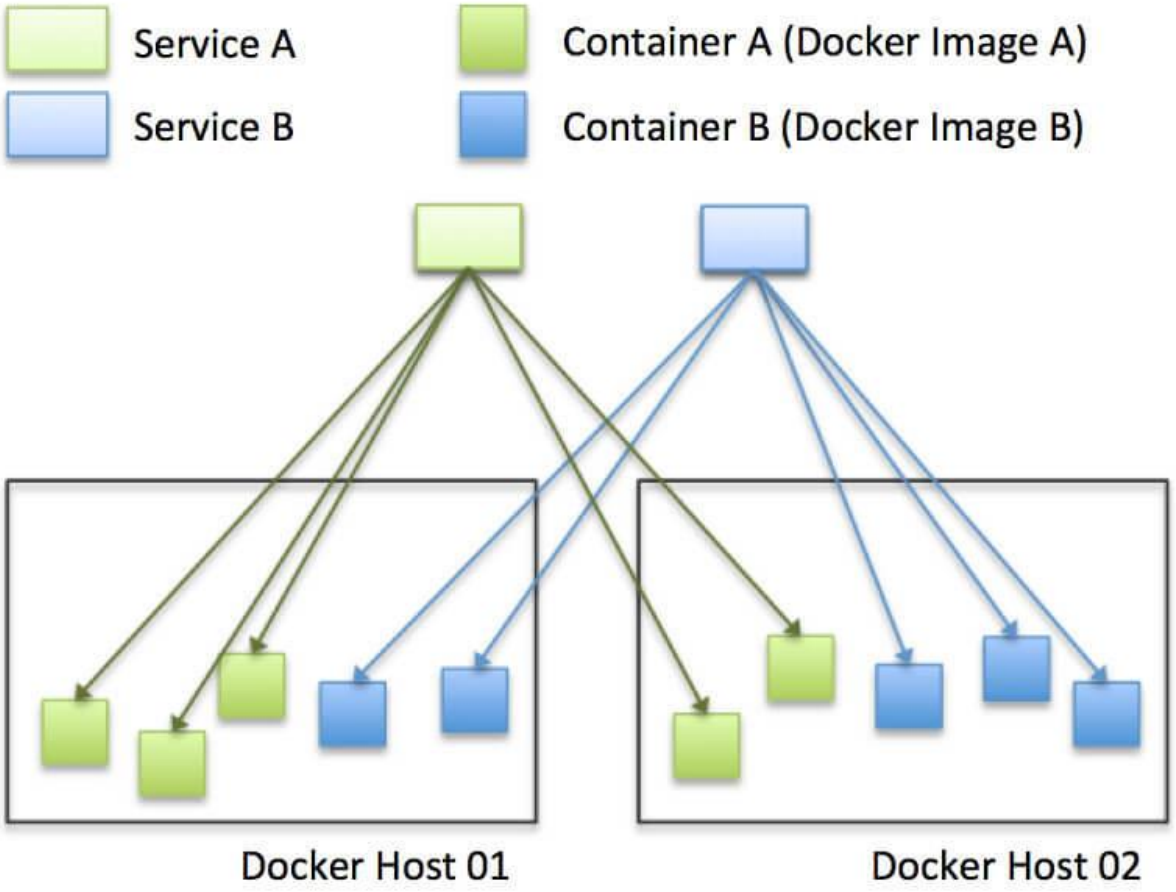



Trường hợp sau khi auto scaling, phát sinh một vấn đề của endpoint đến container. Trong trường hợp sử dụng máy ảo, bằng việc setting load balancing endpoint sẽ được sử dụng như một **VIP**.

**K8s** cũng có một chức năng tương tự như vậy đó là Service. Service của k8s cung cấp chức năng load balancing cho hàng loạt các container được chỉ định. Việc tự động thêm, xoá container thời điểm scale là điều hiển nhiên, khi một container xảy ra sự cố thì tự động cách ly.

Khi thực hiện rolling update container thì việc đầu tiên **k8s** sẽ làm là cách ly container cho chúng ta, vì vậy k8s có thể đảm nhận việc quản lý các endpoint ở mức SLA cao.

Trong kiến trúc Microservice, để sử dụng các image container được tạo ra tương ứng với từng chức năng và deploy chúng thì chức năng Service discovery thực sự cần thiết.




**K8s** là một Platform nhưng có khả năng liên kết tốt với các hệ sinh thái bên ngoài, có nhiều middleware chạy trên các service của k8s, trong tương lai chắc chắn sẽ còn nhiều hơn nữa.

- Ansible: Deploy container tới Kubernetes
- Apache Ignite: Sử dụng Service Discovery của Kubernetes, tự động tạo và scaling k8s clkuster
- Fluentd: gửi log của container trong Kubernetes
- Jenkins: Deploy container đến Kubernetes
- OpenStack ： Cấu trúc k8s liên kết với Cloud
- Prometheus: Monitor Kubernetes
- Spark: Thực thi native job trên Kubernetes(thay thế cho YARN)
- Spinnaker ： Deploy container đến Kubernetes

Thêm nữa, K8s chuẩn bị một vài cơ thế để có thể mở rộng, thực thi chức năng độc lập, nó có thể sử dụng platform như là một framework. Bằng cách sử dụng khả năng mở rộng, chúng ta có thể thực hiện release một ReplicaSet mà k8s cung cấp.

# Những khái niệm cơ bản trong Kubernetes là gì

### Master node

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

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

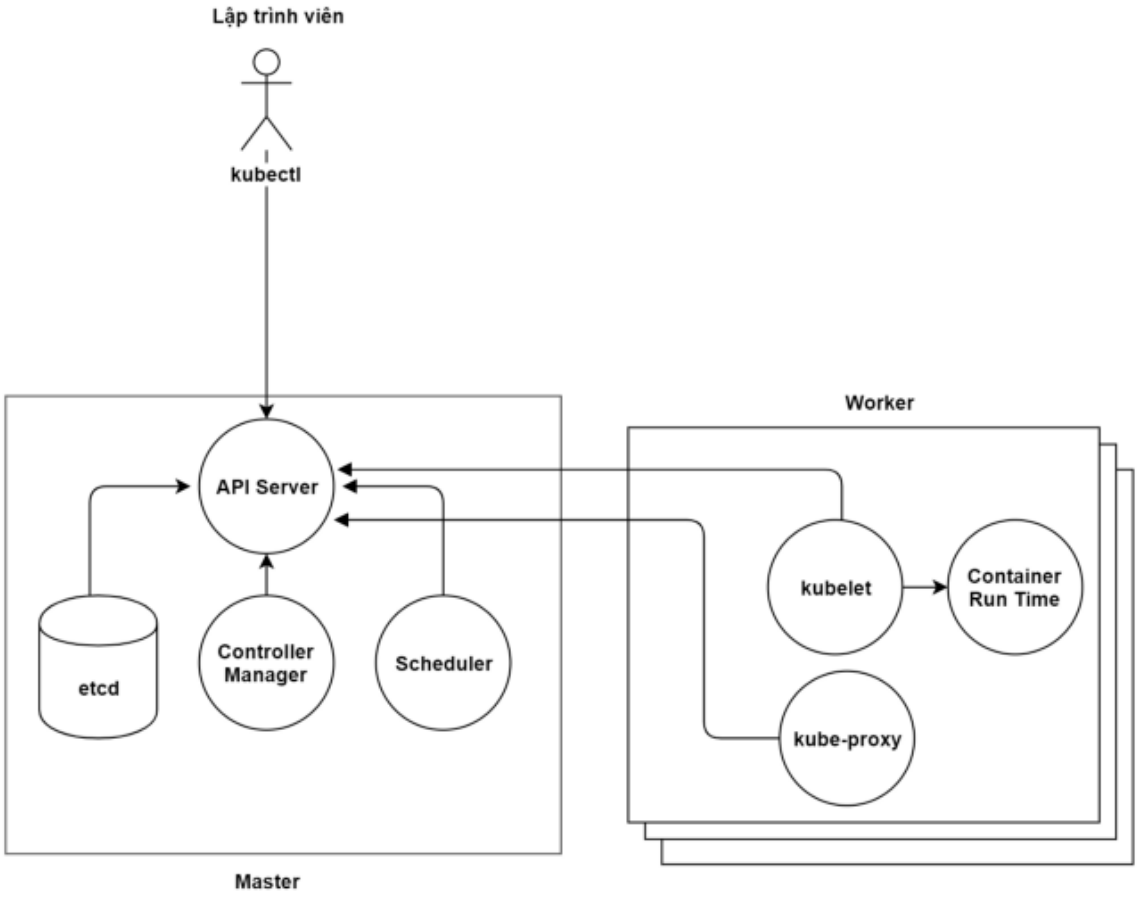
### Worker node

Là server chạy ứng dụng trên đó. Bao gồm 3 thành phần chính:

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

### kubectl

Tool quản trị Kubernetes, được cài đặt trên các máy trạm, cho phép các lập trình viên đẩy các ứng dụng mô tả triển khai vào cụm Kubernetes, cũng như là cho phép các quản trị viên có thể quản trị được cụm Kubernetes.




**Pod**

Pod là khái niệm cơ bản và quan trọng nhất trên Kubernetes. Bản thân Pod có thể chứa 1 hoặc nhiều hơn 1 container. Pod chính là nơi ứng dụng được chạy trong đó. Pod là các tiến trình nằm trên các Worker Node. Bản thân Pod có tài nguyên riêng về file system, cpu, ram, volumes, địa chỉ network…

**Image**

Là phần mềm chạy ứng dụng đã được gói lại thành một chương trình để có thể chạy dưới dạng container. Các Pod sẽ sử dụng các Image để chạy.

Các Image này thông thường quản lý ở một nơi lưu trữ tập trung, ví dụ chúng ta có Docker Hub là nơi chứa Images của nhiều ứng dụng phổ biến như nginx, mysql, wordpress…

**Deployment**

Là cách thức để giúp triển khai, cập nhật, quản trị Pod.

Replicas Controller

Là thành phần quản trị bản sao của Pod, giúp nhân bản hoặc giảm số lượng Pod.

**Service**

Là phần mạng (network) của Kubernetes giúp cho các Pod gọi nhau ổn định hơn, hoặc để Load Balancing giữa nhiều bản sao của Pod, và có thể dùng để dẫn traffic từ người dùng vào ứng dụng (Pod), giúp người dùng có thể sử dụng được ứng dụng.

**Label**

Label ra đời để phân loại và quản lý Pod,. Ví dụ chúng ta có thể đánh nhãn các Pod chạy ở theo chức năng frontend, backend, chạy ở môi trường dev, qc, uat, production…