- cách triển khai, duy trì cơ sở hạ tầng, từ phục hồi, tự mở rộng quy mô trong microservice

- nếu có hàng nghìn instances đang chạy trong ứng dụng doanh nghiệp, làm thế nào để tự động hoá việc triển khai và khôi phục (automate the deployments, rollouts, rollbacks). Như muốn triển khai phiên bản mới của các microservice, và khi triển khai cần không có bất cứ thời gian chết nào

Ex: account microservice với 10 instances, nếu có 1 docker image mới cho account microservice phiên bản tiếp theo chứa các tính năng mới, mong muốn rằng chuyển từ phiên bản cũ sang phiên bản mới

=> thay thể docker image cũ bằng docker image mới bên trong các container sẽ không có khoảng thời gian chết nào.

=> có thể vì lý do bất kỳ, phiên bản mới được triển khai gặp một số vấn đề => làm thế nào để khôi phục tất cả cấc phiên bản trước đó sẽ là 1 thách thức, đảm bảo thực hiện thách thức này cho tất cả các microservice 1 các tức thì và không cần triển khai thủ công (mong muốn tự động hoá)

- vấn đề về tự động hồi phục các service (make services are self-healing)

Ex: nếu trong microservice, 1 trong những instance không phản hồi, vị trí địa lý nới triển khai gặp 1 số sự cố mạng. Do đó không thể cung cấp bất kỳ phản hồi nào cho khách hàng

=> cần 1 cơ chế để tự động giúp kill container và mang đến 1 container mới tại đó (không cần giám sát từng microservice thủ công)

- làm thế nào mở rộng quy mô dịch vụ (auto scale service)

Ex: netflix sẽ có lượng truy cập ít hơn vào thứ 2 - 6, và nhiều hơn vào t7, cn

=> netflix nên có 1 cơ chế để xác định tải của microservice dựa trên các số liệu như mức sử dụng cpu và có cách tự động mở rộng quy mô bằng các thêm nhiều instance hơn

==========================================================================================================================

Kubernetes: là 1 hệ thống mã nguồn mở cho việc tự động triển khai, mở rộng và quản lý các containerzed applications. Đây là nền tẳng dàn dựng nổi tiếng nhất và nó là nền tảng cloud neural. (còn được viết tắt là K8s)

Kubernetes cung cấp cho bạn với 1 khuôn khổ để chạy các dịch vụ phân tán 1 cách bền bỉ. Nó đảm nhận việc mở rộng quy mô và chuyển đổi dự phòng cho ứng dung, cung cấp các mẫu triển khai và hơn thế nữa:

- Service discovery và load balancung (phát hiện và cân bằng tải)

- Storage orchestration (điều phối lưu trữ)

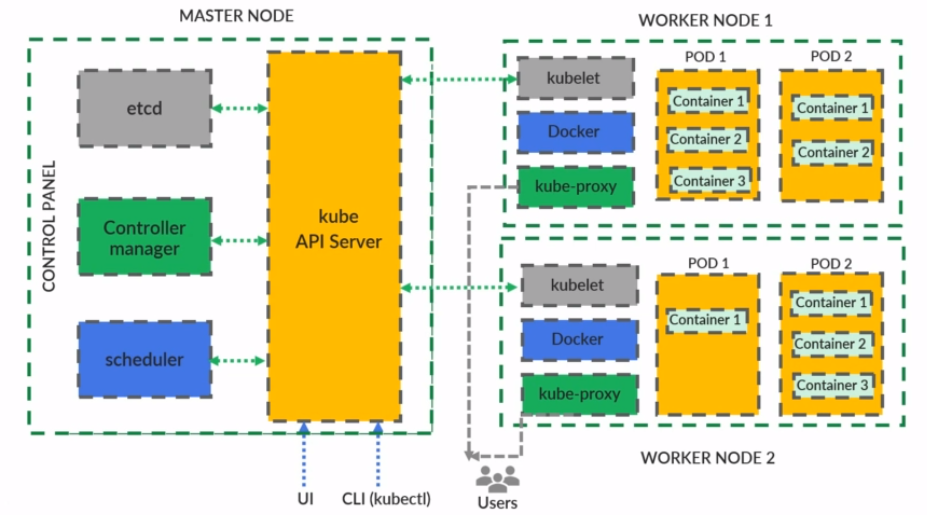
- Tự động rollouts (triển khai) và rollback(quay lại phiên bản cũ)

- Automatic bin packing

- Self-healing (tạo instance mới thay instance hỏng)

- Secret và configuration management (nội bộ kubernetes có cơ chế xử lý các bí mật và các thuộc tính bằng cách sử dụng quản lý cấu hình bên trong nó)

Kiến trúc bên trong của kubernetes:



- Về cơ bản, kubernetes sẽ xử lý các cluster của node trong kiến trúc microservice

- cluster là tập hợp rất nhiều server khác nhau làm việc cùng nhau để đạt được kết quả. Cluster chứa nhiều server cần phải có 1 tiêu chuẩn để duy trì các server trong cluster

=> cluster có 1 node master và các node worker, các node worker đảm nhận tất cả công việc như nhận request của client, thực thi code trong container hoặc docker image. Trong khi đó node master sẽ chăm sóc toàn bộ cluster để chạy khoẻ mạnh không gặp bất kỳ vấn đề gì, nếu có vấn đề, master sẽ tự phục hồi để cluster luôn trong tình trạng khoẻ mạnh.

=> Khi có hàng nghìn microservice và các instance của nó, chúng ta triển khai tất cả các microservice docker container và các server khác nhau trên toàn cầu (1 vài cho ng châu á, 1 vài cho ng châu âu, ..). Bất kể triển khai ở đâu, chúng cũng đều là node worker từ 1 cluster, và master node sẽ quản lý chúng, có thể có hàng nghìn node worker. Chúng ta cũng có thể có 1 số node master, dựa trên số lượng node worker cần quản lý.

- vai trò và thành phần của master node

- vai trò đảm bảo tất cả woker node đang hoạt động tốt mà không có bất kỳ vấn đề nào bên trong cluster

- có 4 thành phần quan trong trong master node;

1. Kube API Server: giống như bất kỳ rest service, master node sẽ hiển thị tất cả các api sử dụng kube APi Server. Bất cứ ai muốn tương tác với cluster từ bên ngoài, họ cần phải đi qua kube api server. Có thể nghĩ kube api server như gateway của cluster, nó chỉ cho những người dùng được xác thực hoặc hệ thống được xác thực mới tương tác được với cluster. Bất cứ khi nào muốn tương tác với kubernetes cluster, chỉ có thể tương tác với master node, không bao giờ tương tác được với worker node. Tương tác với kube API Server có thể qua UI hoặc CLI (kubectl).

2. Scheduler: là thành phần chịu trách nhiệm làm việc với các node worker, làm tất cả các tính toán nội bộ: Ex: nếu muốn tạo 1 container mới dựa trên microservice mới, nó sẽ đi kiểm tra xem worker node nào có ít container, dựa trên đó sẽ lên lịch triển khai vào worker tương ứng. Dựa trên hướng dẫn lên lịch đến các worker, container sẽ được triển khai trong 1 port.

=> Scheduler sẽ lập lịch cho việc thêm, xoá, cập nhật của tất cả các container bằng cách nhận các lệnh từ kube API Server (UI hoặc CLI)

3. Controller manager kiểm tra sức khoẻ của từng container, nếu gặp vấn đề sẽ thay thế nó bằng container mới. Dựa trên định nghĩa đã đưa ra trong quá trình triển khai, nó sẽ luôn cố gắng đạt được mục tiêu chúng ta đề ra.

=> có design state và current state, controller manager sẽ luôn đảm bảo desgin state và curent state sẽ khớp nhau

1. Etcd: là bộ não của cluster, là cơ sở dữ liệu lưu trữ thông tin các tất cả các chi tiết cluster (bao nhiêu worker node, bao nhiêu phần trong từng worker node, có bao nhiêu container được triển khai…). Bất cứ khi nào scheduler tạo một port mới hoặc 1 worker mới, nó phải đến cập nhật trên etcd. Etcd cũng lưu trữ những command của người dùng (giửu qua kube api server), controller manager tương tác với etcd để hiểu những gì được mong đợi.

=> 4 thành phần này gọi là control panel. Master node chỉ quản lý các worker node container với sự hỗ trợ của worker node trong cluster. Nó không cần nhiều tài nguyên máy chủ. Do đó, master node được triển khai với dung lương ít so với worker node.

Worker node (10p45 [Master Microservices with Java, Spring, Docker, Kubernetes | Udemy](https://www.udemy.com/course/master-microservices-with-spring-docker-kubernetes/learn/lecture/27185122" \l "content) )