BAN CƠ YẾU CHÍNH PHỦ

HỌC VIỆN KỸ THUẬT MẬT MÃ

ĐỀ CƯƠNG

CHUYÊN ĐỀ AN TOÀN HỆ THỐNG THÔNG TIN

Nghiên cứu giải pháp đảm bảo an toàn dữ liệu trên Kubernetes

Ngành: An toàn thông tin

Sinh viên thực hiện:
Nguyễn Anh Tuấn
Mã sinh viên: AT160258
Đặng Sơn Hà
Mã sinh viên: AT160220

Người hướng dẫn:
TS. Nguyễn Mạnh Thắng
Khoa An toàn thông tin - Học viện Kỹ thuật mật mã

Hà Nội, 2022

Mục lục

Lời cảm ơn

Nhóm chúng em xin chân thành cảm ơn các thầy cô trường Học viện Kỹ thuật Mật mã nói chung, quý thầy cô của khoa An toàn thông tin nói riêng đã tận tình dạy bảo, truyền đạt kiến thức cho chúng em trong suốt quá trình học.

Kính gửi đến Thầy Nguyễn Mạnh Thắng lời cảm ơn chân thành và sâu sắc nhất, cảm ơn thầy đã tận tình theo sát, chỉ bảo và hướng dẫn cho nhóm em trong quá trình thực hiện đề tài này. Thầy không chỉ hướng dẫn chúng em những kiến thức chuyên ngành, mà còn giúp chúng em học thêm những kĩ năng mềm, tinh thần học hỏi, thái độ khi làm việc nhóm.

Trong quá trình tìm hiểu nhóm chúng em xin cảm ơn các bạn sinh viên đã góp ý, giúp đỡ và hỗ trợ nhóm em rất nhiều trong quá trình tìm hiểu và làm đề tài.

Do kiến thức còn nhiều hạn chế nên không thể tránh khỏi những thiếu sót trong quá trình làm đề tài.Chúng em rất mong nhận được sự đóng góp ý kiến của quý thầy cô để đề tài của chúng em đạt được kết quả tốt hơn.

Chúng em xin chân thành cảm ơn!

Lời mở đầu

Trong những năm gần đây, Kubernetes đã trở thành một trong những công cụ quản lý container phổ biến nhất. Điều này là do sự phát triển mạnh mẽ của Kubernetes, cũng như sự phổ biến của container. Trong các doanh nghiệp hiện nay, Kubernetes là một phần không thể thiếu được sử dụng để quản lý các ứng dụng container, cung cấp các dịch vụ như load balancing, autoscaling, logging, monitoring, backup và restore, cũng như quản lý các tài nguyên của các ứng dụng như CPU, Memory, Storage, Network. Tuy nhiên, việc sử dụng Kubernetes cũng có những hạn chế, đặc biệt là trong việc quản lý các cấu hình nhạy cảm của ứng dụng và các tài nguyên của ứng dụng. Vì vậy, trong đề tài này, chúng ta sẽ tìm hiểu về Vault, một công cụ quản lý secret phổ biến nhất hiện nay, được phát triển bởi HashiCorp.

Vault được sử dụng để quản lý các secret như username, password, token, key, certificate, API key, SSH key, license key, database connection và các thông tin khác. Vì vậy trong tương lai, Vault sẽ là một trong những công cụ quản lý secret không thể thiếu trong Kubernetes.

Với những lí do trên, bài toán đặt ra ở đây là làm sao để triển khai Vault trên cụm Kubernetes. Vậy nên, đề tài "Nghiên cứu giải pháp đảm bảo an toàn dữ liệu trên K8S" được thực hiện trong báo cáo có ý nghĩa khoa học và mang tính thực tiễn cao.

Chương 1

Giới thiệu về Kurbenetes

1.1 Tổng quan về Kurbenetes

Kubernetes hoặc k8s là một nền tảng mã nguồn mở giúp tự động hóa việc quản lý, mở rộng và triển khai ứng dụng dưới dạng container. Kubernetes còn được gọi là Container Orchestration Engine (hiểu nôm na là công cụ điều phối container). Kubernetes loại bỏ rất nhiều các quy trình thủ công liên quan đến việc triển khai và mở rộng các containerized applications. Các ứng dụng có thể khác nhau về kích thước: từ 1 cho đến hàng nghìn server. Với Kubernetes chúng ta có thể phát triển application một cách linh hoạt và đáng tin cậy.

Trách nhiệm chính của Kubernetes là container orchestration (dịch ra có nghĩa điều phối container). Kubernetes đảm bảo rằng tất cả container được lên lịch chạy trên các cụm server (cluster) (server ở đây có thể là physical machine hoặc virtual machine). Ngoài ra, Kubernetes còn có chức năng theo dõi hoạt động của từng container, mở rộng các containers và quản lý tình trạng của các containers theo thời gian. Khi một container nào đó gặp trục trặc, dừng hoạt động thì Kubernetes sẽ thay thế container đó.

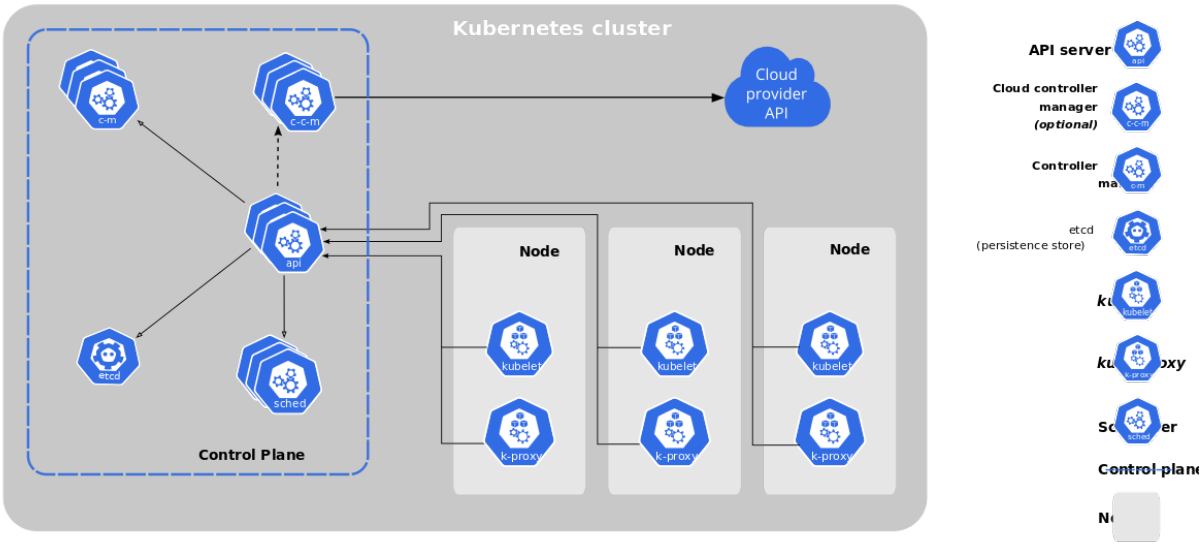
Kubernetes ban đầu được các kỹ sư Google phát triển vào năm 2014. Và Google cũng là một trong những cái tên tiên phong đóng góp cho công cuộc phát triển công nghệ Linux container.

Kubernetes ban đầu được các kỹ sư Google phát triển vào năm 2014. Và Google cũng là một trong những cái tên tiên phong đóng góp cho công cuộc phát triển công nghệ Linux container.

Red Hat là một trong những công ty đầu tiên hợp tác với Google trong dự án Kubernetes trước khi Kubernetes được ra mắt, và trở thành nhà tài trợ lớn thứ 2 cho dự án ngay từ những ngày đầu. Và Google đã tặng lại dự án Kubernetes cho Cloud Native Computing Foundation (CNCF - thành lập năm 2015). Kubernetes đã trở thành một project của tổ chức này cho đến thời điểm hiện tại. Năm 2018, Kubernetes Project đạt vị trí thứ 9 về số lượt commit trên Github.

1.1.1 Các thành phần trong kubernetes

Khi bạn triển khai Kubernetes, bạn sẽ nhận được một Cluster (cụm máy chủ).



Hình 1.1: Sơ đồ của một cụm Kubernetes.

Một Cluster Kubernetes bao gồm một tập hợp các máy worker, được gọi là Node, chạy các ứng dụng được đóng gói. Mỗi Cluster có ít nhất một Node.

(Các) Node lưu trữ các Pod - Một nhóm gồm một hoặc nhiều containers được triển khai cho single Node, nhóm này chia sẻ chung không gian lưu trữ, IP address, hostname, và những nguồn khác.

Các Control Plane quản lý các Node và các Pod trong Cluster. Trong môi trường production, Control Plane thường chạy trên nhiều máy tính và một Cluster thường chạy nhiều Node, mang lại khả năng chịu lỗi và tính sẵn sàng cao.

Trên đây là sơ đồ của một cụm Kubernetes với tất cả các thành phần được gắn với nhau.

Các thành phần Control Plane (Master)

Các thành phần Control Plane đưa ra các quyết định chung về Cluster (ví dụ: lập lịch), cũng như phát hiện và phản hồi các sự kiện của Cluster.

Các thành phần Control Plane có thể được chạy trên bất kỳ máy nào trong Cluster. Tuy nhiên, để đơn giản, thiết lập tập lệnh thường khởi động tất cả các thành phần Control Plane trên cùng một máy và không chạy các container của người dùng trên máy này.

Đối với cluster nhỏ , Master có thể chạy trên một Node, nhưng trong một cluster lớn, để đảm bảo tính khả dụng (trong tiếng anh là High-Availability) thì Master có thể được chạy trên nhiều Node. (Tính khả dụng có nghĩa là Khi mà một Node trong cluster dừng hoạt động thì hệ thống vẫn duy trì như không có gì xảy ra). Master sẽ bao gồm 5 thành phần chính sau:

- **kube-apiserver:**

  API Server là một thành phần của Control Plane. Đúng theo tên gọi, đây chính là server cung cấp REST API cho Kubernetes Cluster. Nó có nhiệm vụ đặt Pod vào

Node, đồng bộ hoá thông tin của Pod bằng REST API tiếp nhận cài đặt, xác thực và thiết lập cho các objects pod/service/replicationController.

- **Etcd:**

  Etcd là Kho lưu trữ giá trị khóa nhất quán và có tính khả dụng cao được sử dụng làm kho dự phòng để lưu trữ toàn bộ cấu hình, trạng thái và metadata của Kubernetes Cluster.

  Trong các cluster nhỏ, Etcd có thể chạy trên cùng một Node với các thành phần khác. Nhưng trong các cluster lớn, Etcd có thể chạy dự phòng trên nhiều Node để đảm bảo tính khả dụng của toàn hệ thống. Nếu cụm Kubernetes của bạn sử dụng Etcd làm nơi lưu trữ sao lưu, hãy đảm bảo rằng bạn có kế hoạch sao lưu cho những dữ liệu đó.

- **kube-controller-manager:**

  Kube Controller Manage một tập hợp các controller khác nhau để chạy các bộ điều khiển quy trình theo dõi các cập nhật trạng thái của Kubernetes Cluster thông qua API và thực hiện các thay đổi đối với Cluster sao cho phù hợp.

  Về mặt logic, mỗi bộ điều khiển là một quy trình riêng biệt, nhưng để giảm độ phức tạp, tất cả chúng đều được biên dịch thành một tệp nhị phân duy nhất và chạy trong một quy trình duy nhất.

  Một số loại bộ điều khiển có thể kể đến:

  - Node controller: Chịu trách nhiệm thông báo và phản hồi khi các Node gặp sự cố.
  - Job controller: Kiểm soát các tác vụ được lập lịch, sau đó tạo các Pod để chạy các tác vụ đó đến khi hoàn thành.
  - Endpoints controller: Điền khiển đối tượng Điểm cuối (nghĩa là tham gia Dịch vụ & Nhóm).
  - Service Account & Token controllers: Tạo tài khoản mặc định và mã thông báo truy cập API cho không gian tên mới.

- **cloud-controller-manager:**

  Là một tập hợp các logic dành riêng cho đám mây (GCP, AWS, Azure) cho phép bạn liên kết Kubernetes Cluster với API của nhà cung cấp đám mây và tách các thành phần tương tác với nền tảng đám mây đó khỏi các thành phần chỉ tương tác với cụm của bạn. Cloud-controller-manager chỉ chạy các bộ điều khiển dành riêng cho nhà cung cấp dịch vụ đám mây của bạn. Nếu bạn đang chạy Kubernetes tại cơ sở của riêng mình hoặc trong môi trường học tập bên trong PC của riêng bạn, cụm không có trình quản lý bộ điều khiển đám mây.

  Cũng như với kube-controller-manager, cloud-controller-manager kết hợp một số vòng điều khiển độc lập về mặt logic thành một tệp nhị phân duy nhất mà bạn chạy như một quy trình duy nhất. Bạn có thể chia tỷ lệ theo chiều ngang (chạy nhiều hơn một bản sao) để cải thiện hiệu suất hoặc để giúp tăng khả năng chịu lỗi.

  Các bộ điều khiển sau có thể có các phần phụ thuộc của nhà cung cấp dịch vụ đám mây:

  - Node controller: Để kiểm tra nhà cung cấp đám mây để xác định xem một Node đã bị xóa trong đám mây sau khi nó ngừng phản hồi hay chưa.

- Route controller: Để thiết lập định tuyến trong cơ sở hạ tầng đám mây cơ bản.
- Service controller: Để tạo, cập nhật và xóa bộ cân bằng tải của nhà cung cấp dịch vụ đám mây.

- **kube-scheduler:**

  Sử dụng Kubernetes API để tìm, theo dõi các Pod chưa được lên lịch hoặc mới được tạo nhưng không được chỉ định Node. Sau đó, scheduler sẽ đặt các Pod này vào các Node dựa trên tài nguyên và các ràng buộc khác được định nghĩa trong manifest file của Pod. Scheduler sẽ cố gắng đảm bảo rằng các Pod của cùng một application sẽ được phân phối trên các Node khác nhau để đảm bảo tính khả dụng.

  Các yếu tố được tính đến để đưa ra quyết định lập lịch bao gồm: yêu cầu tài nguyên cá nhân và tập thể, các ràng buộc về phần cứng / phần mềm / chính sách, thông số kỹ thuật về mối quan hệ và chống mối quan hệ, vị trí dữ liệu, can thiệp giữa khối lượng công việc và thời hạn.

Các thành phần Worker

Có nhiệm vụ xử lý khối lượng công việc của application trong cluster, duy trì các nhóm đang chạy và cung cấp môi trường runtime cho Kubernetes. Worker sẽ bao gồm 3 thành phần chính sau:

- **kubelet:**

  Kubelet chạy trên mỗi Worker Node. Nó đảm bảo rằng các container đang chạy trong một Pod và có trách nhiệm giám sát giao tiếp với master node và quản lý các Pod. Kubelet sử dụng CRI (Container Runtime Interface - Giao diện thời gian chạy vùng chứa) để giao tiếp với container runtime trên cùng một Node đó. Kubelet không quản lý các vùng chứa không được tạo bởi Kubernetes.

- **kube-proxy:**

  Là một proxy mạng chạy trên mỗi Node trong Cluster. Nó có trách nhiệm quản lý, duy trì các quy tắc mạng trên mỗi Node. Các quy tắc mạng này cho phép giao tiếp mạng với Pod của bạn từ các phiên mạng bên trong hoặc bên ngoài Cluster. kube-proxy sử dụng lớp lọc gói của hệ điều hành nếu có và nó có sẵn. Nếu không, kube-proxy sẽ tự chuyển tiếp lưu lượng.

- **Container Runtime:**

  Là phần mềm chịu trách nhiệm chạy các container. Kubernetes hỗ trợ một số container runtime như: Docker, containerd, CRI-O và bất kỳ triển khai nào của Kubernetes CRI (Container Runtime Interface).