**DOCKER SWARM – WITH PLURALSIGHT**

# Tổng quan

Bắt đầu với **Docker** **Swarm**. Dễ dàng chạy bất kỳ ứng dụng nào mà bạn có thể nghĩ ra với **Docker**, kéo một image, tạo volume. Tuy nhiên, sẽ có lúc project làm cạn kiệt tài nguyên trên một máy tính và sẽ không tốt nếu ta có thể sử dụng cùng một workflow để tạo ra các application trên toàn bộ một cụm máy? Trong khóa học này, chúng ta sẽ xem cách **Docker** **Swarm** làm cho điều này trở nên khả thi. Chúng ta sẽ bắt đầu bằng cách tạo một cụm node. Cùng đi sẽ xem xét việc triển khai các service hoặc application trên các node đó. Chúng tôi sẽ xem xét các stack để giúp chúng tôi quản lý không chỉ các service mà còn cả network, volume, config, tất cả các resource cần thiết để chạy một application. Chúng tôi sẽ xem xét cách chúng tôi cập nhật ứng dụng của mình với **Docker** **Swarm**. Chúng tôi thậm chí sẽ xem xét một trong những tính năng mới nhất, công việc. Vào cuối khóa học này, bạn sẽ chuẩn bị cung cấp một cụm và triển khai các ứng dụng phức tạp cho nó. Đến với khóa học này, tôi sẽ giả sử rằng bạn đã quen thuộc với một node **Docker** độc lập sử dụng **Docker** CLI và có lẽ thậm chí là **Docker** Compose. Được rồi, bắt đầu thôi.

# Tham gia các node để tạo thành một cụm Docker Swarm

Từ Standalone đến multi-node

**Docker** là một trong những công cụ yêu thích của tôi từ trước đến nay. Và một phần lớn của điều đó là vì tôi có thể nhớ lịch sử nơi chúng tôi có một máy duy nhất, đôi khi là máy của nhà phát triển, lúc khác là server, chúng tôi có phần cứng và hệ điều hành của chúng tôi và chúng tôi có các process chạy trên máy đó. Tôi có thể nhớ nỗi đau của việc triển khai các bản cập nhật và xử lý các tình huống như nhiều ứng dụng cần truy cập mạng trên cùng một cổng, trừ khi có thể chúng đã được định cấu hình để sử dụng một cổng khác và đó là điều tôi phải lo lắng. Có thể họ có thể được, có thể họ không thể. Có thể tôi đã biết cách làm, có thể tôi đã không và phải dành một thời gian để tìm ra nó. Hoặc, có thể hai ứng dụng khác cần quyền truy cập vào thư viện được chia sẻ và một trong số chúng ghi đè lên các ứng dụng khác và do đó ứng dụng khác sẽ chết, vì vậy chúng tôi phải tìm ra vấn đề này, sau đó thử và cô lập các hệ thống tệp mà cả hai các ứng dụng đang sử dụng, hãy cung cấp cho chúng các thư mục riêng, ngăn chúng làm việc của nhau và hy vọng rằng các thư viện hoặc khuôn khổ được chia sẻ này không cần phải tồn tại ở một nơi cụ thể. Ví dụ: nếu chúng ta cần các phiên bản khác nhau của chúng, có thể là các phiên bản khác nhau của thời gian chạy Java hoặc thời gian chạy .NET cũ hơn không thể tồn tại song song với nhau. Và tất nhiên, các volume đã giúp chúng tôi khắc phục sự cố này bằng cách lấy một máy duy nhất và cắt các tài nguyên sao cho giống như chúng tôi có các máy riêng lẻ cho từng ứng dụng nếu chúng tôi muốn hoặc từng tập hợp các ứng dụng có liên quan chặt chẽ với nhau. Nó không thực sự là một máy ảo, nhưng nó là một lớp ảo hóa trông gần như giống nhau bằng cách cô lập những gì mà một tiến trình có quyền truy cập. Bạn có thể cô lập các card mạng để mỗi vùng chứa, nó xuất hiện như thể nó có card mạng riêng với địa chỉ IP riêng, ngay cả trên mạng công cộng, không có xung đột cổng nào trong trường hợp đó. Hoặc, hai ứng dụng đó có thư viện được chia sẻ, mỗi ứng dụng có thể có bản sao của riêng mình, chúng có thể sống ở cùng một vị trí chính xác trên đĩa và không có vấn đề gì vì mỗi ứng dụng có hệ thống tệp riêng hoặc có vẻ như nó có hệ thống tệp riêng. hệ thống tập tin, gắn kết của riêng nó. Tất nhiên, cũng có những lợi ích khác, tôi không chỉ là sự cô lập. Tôi nghĩ lý do lớn khiến **Docker** thành công như vậy là do khái niệm đăng ký này, **Docker** Hub, phân phối các image dựng sẵn chứa ít nhiều hệ thống tệp cho bất kỳ ứng dụng nhất định nào. Và điều thực sự khiến **Docker** phát triển là thực tế là rất nhiều người đã tham gia và bạn có thể tìm thấy hầu hết mọi ứng dụng liên quan đến lập trình phía máy chủ , bất kỳ ứng dụng nào trên **Docker** Hub dưới dạng image chính thức mà bạn có thể tin tưởng, các ứng dụng như MongoDB hoặc NGINX hoặc MySQL, hoặc thậm chí một cái gì đó như Jenkins, cả máy chủ và tác nhân trong trường hợp đó. Các image có sẵn. Chúng được cung cấp chính thức bởi **Docker** và nhóm phát triển chúng hoặc nhóm tự phát triển phần mềm. Và vì vậy, chúng tôi không chỉ cung cấp cho mỗi vùng chứa hệ thống tệp riêng của nó, mà những hệ thống tệp đó luôn được tạo sẵn và có sẵn cho chúng tôi. Bây giờ nếu bạn giống tôi, bạn sẽ không mất nhiều thời gian từ việc tự hỏi **Docker** là gì để tìm cách tận dụng lợi thế của nó trong môi trường phát triển của riêng bạn bởi vì đó là một trong những nơi tốt nhất để bắt đầu khi học về **Docker**. Có những lợi ích được gặt hái mà thậm chí không cần phải mạo hiểm với bất kỳ loại môi trường sản xuất nào hoặc thậm chí là môi trường thử nghiệm. Có nhiều năm lợi ích chỉ trong phát triển một mình hoặc máy trạm cá nhân của bạn, thậm chí chỉ là một máy tính ở nhà mà bạn sử dụng. Ví dụ, tôi thấy mình đang điều hành TeamCity để thực hiện các cuộc biểu tình. Khi một phiên bản mới ra mắt, tôi chỉ cần kéo các image mới xuống, chúng sẽ được thiết lập và chạy, và tắt tôi đi với phần trình diễn các tính năng. Và đêm trước khi tôi có thể thăm dò xung quanh, xem những gì đã thay đổi, nếu có bản beta, tôi cũng có thể kéo chúng xuống. Tôi thậm chí có thể chạy các đại lý như thể chúng là các node riêng biệt mô phỏng các cụm. Tôi có thể trình diễn rất nhiều node liên quan. Vì vậy, tôi có thể giới thiệu các tính năng hướng theo cụm , điều này rất khó thực hiện trong một bản demo truyền thống. Và sau đó, có thể đằng sau hậu trường, hoặc ít nhất là tạm dừng để trình diễn TeamCity, tôi cũng có thể có một ứng dụng .NET mà tôi đang phát triển và ứng dụng .NET đó có thể sử dụng cơ sở dữ liệu Mongo hoặc có thể là Maria DB, MySQL, thậm chí là Microsoft SQL Server hiện có sẵn dưới dạng image và không chỉ dành cho Windows, nó cũng có sẵn với Linux. Trên thực tế, tôi có thể nói rằng **Docker** là một trong những lý do đã kéo hoặc thúc đẩy Microsoft theo hướng trở nên thân thiện hơn với Linux. Và với tất cả các ứng dụng web này trên máy của tôi, tôi có thể muốn một số loại proxy có liên quan để tôi có thể xử lý các yêu cầu từ thế giới bên ngoài nếu tôi có môi trường trình diễn hoặc thậm chí là môi trường sản xuất và mở nó lên mạng cục bộ của tôi hoặc bằng cách khác, và Tôi có thể sử dụng thứ gì đó ngoài NGINX. Có rất nhiều lựa chọn thay thế trên **Docker** Hub. Vào cuối ngày, những gì chúng tôi chạy ít liên quan hơn thực tế là chúng tôi có rất nhiều volume, và may mắn thay, chúng tôi có những cỗ máy to khỏe ngày nay nên chúng tôi có thể chạy rất nhiều chúng. Nhưng điều đó không có nghĩa là tại một thời điểm nào đó, tài nguyên sẽ không bị giới hạn trên máy đó. Khi chúng ta kéo ngày càng nhiều vùng chứa xuống, rất dễ hết dung lượng đĩa, chẳng hạn như chỉ cần lưu trữ tất cả những image đó và may mắn thay, những image đó được lưu trữ theo cách để giảm dung lượng ổ đĩa bằng cách sử dụng phân lớp. Tuy nhiên, việc bạn có thể tạo ra các phiên bản khác nhau của bất kỳ ứng dụng nào dễ dàng hơn cũng khiến bạn muốn giữ nhiều bản sao xung quanh. Và tất nhiên, chúng tôi có thể phát triển máy của mình lớn hơn, và điều này sẽ có hiệu quả và có thể sẽ hoạt động lâu dài đối với máy phát triển, nhưng ngay cả đối với phát triển, và đặc biệt là đối với môi trường sản xuất và thử nghiệm nơi bạn có nhiều tải, một chiếc máy duy nhất sẽ không cắt được nó. Và ngoài lý do về tài nguyên, còn vì lý do về độ tin cậy, chuyển đổi dự phòng, tính khả dụng, thậm chí là khả năng truy cập vào các nền tảng khác nhau bởi vì những ý tưởng giúp nó có thể chứa một ứng dụng và phân phối nó với một image, tốt, điều đó sẽ không là một điều thực sự tuyệt vời nếu chúng ta có thể sử dụng nó ngoài một chiếc máy duy nhất? Tất nhiên, chúng tôi có thể cài đặt **Docker** trên nhiều máy và nếu bạn giống tôi lúc đầu, bạn có thể đã thử điều này, nơi bạn có thể điều khiển từ xa vào từng máy này và kiểm soát những gì đang chạy trên chúng. Đó là một cách tiếp cận khá khả thi khi bắt đầu, nhưng khi nhu cầu của bạn ngày càng lớn hơn về tài nguyên và tính khả dụng cũng như chuyển đổi dự phòng, việc nhảy vào từng hộp không thú vị lắm, ngay cả với các tính năng ngữ cảnh **Docker** mới nhất giúp bạn dễ dàng chuyển ngữ cảnh sang từng điểm ra các cài đặt **Docker** khác nhau. Sự thật là sẽ rất tuyệt nếu chúng ta có thể coi đây là một cỗ máy khổng lồ, ngay cả khi có rất nhiều người tham gia. Ý tôi là, sẽ không tốt nếu số lượng máy hoàn toàn không liên quan như bạn nghĩ? Và, tất nhiên, việc nâng nặng phải được thực hiện ở hậu trường để làm cho điều đó có thể thực hiện được, nhưng điều đó sẽ khiến nó thực sự tuyệt vời đối với người dùng cuối. Và có lẽ điều giá trị nhất về **Docker** là làm cho mọi thứ trở nên dễ dàng đối với những người sử dụng nó, cho dù bạn là nhà phát triển, người thử nghiệm, ai đó hỗ trợ và vận hành một ứng dụng, các công cụ mà họ đã sản xuất luôn thực hiện rất tốt quét dưới tấm thảm sự phức tạp, mọi thứ diễn ra như thế nào, để những gì là rất rõ ràng và nó khá dễ dàng để làm việc một khi bạn có thể.

# Khởi tạo một bầy node đơn với Docker Swarm Init

Khi nói đến phân cụm với **Docker** **Swarm**, tôi thấy rằng nó hữu ích và điều này cũng sẽ áp dụng cho các công cụ phân cụm khác, sẽ rất hữu ích khi tách ra hai khái niệm. Chúng tôi có vùng chứa của chúng tôi mà chúng tôi đã đơn giản hóa, và sau đó chúng tôi cũng có node, bất kể những gì đang chạy trên đó cho một hệ điều hành, bất kể nó có phần cứng gì. Vì vậy, chúng tôi muốn giữ hai khái niệm riêng biệt này được tách biệt ở một mức độ. Mỗi phần trong số này đều có đơn giản hóa quá mức, nhưng mỗi phần trong số chúng đại diện cho một vòng đời quản lý riêng biệt. Chúng ta phải đi về quá trình tạo các node, cài đặt **Docker** trên chúng và sau đó chúng ta có thể tiếp tục quá trình kết hợp các node này lại với nhau để chúng ta có thể bắt đầu tận dụng các tính năng của **Docker** **Swarm** cho phép chúng ta nghĩ đến mọi thứ như một máy lớn, một node lớn chứa tất cả các volume của chúng tôi hoặc bất kỳ tài nguyên nào chúng tôi muốn, chạy trên đó. Vì vậy, tại sao chúng ta không làm điều đó? Hãy tiếp tục và đi về quá trình tạo ra bầy đàn đầu tiên của chúng ta. Vì vậy, chúng ta hãy sao lưu ở đây một chút thời gian. Chúng tôi có ba node khái niệm riêng biệt. Và một trong những cách dễ nhất để bắt đầu thực sự là chỉ cần quên hai node còn lại và tập trung vào một cụm node duy nhất. Sau cùng, bạn phải bắt đầu với một node trước khi các node khác có thể tham gia vào nó. Vì vậy, nếu chúng ta tạo một cụm gồm một node, chúng ta có thể làm việc với rất nhiều giao diện của **Docker** **Swarm**, đặc biệt là các lệnh CLI, mà không có sự phức tạp của nhiều máy liên quan. Vì vậy, đối với trường hợp này, chúng ta chỉ cần tìm một node duy nhất đã được cài đặt **Docker** trên đó. Điều đó có thể ở đâu đối với hầu hết những người đã quen thuộc với **Docker**? Với sự phổ biến của Windows và Mac OS, các bản dựng **Docker** Desktop cho mỗi hệ điều hành này là một nơi tuyệt vời để bắt đầu. Sau khi cài đặt **Docker**, bạn chỉ cần chuyển đến dòng lệnh và chúng ta có thể chạy một lệnh có tên là thông tin **docker**. Bên dưới phía máy chủ, có một phần **Swarm** và nó được gắn nhãn là không hoạt động. Bạn có thể đã nghe nói về **Docker** **Swarm** được gọi là chế độ **Docker** **Swarm**, và đó là bởi vì nó giống như một node chuyển đổi mà bạn có thể bật và tắt. Tất cả những gì bạn phải làm là chạy lệnh **Swarm** của **docker** và tôi sẽ không thực sự chạy lệnh con cần thiết để làm điều này, tôi muốn cho bạn thấy có một danh sách các lệnh có sẵn và bên trong danh sách đó, bạn có thể thấy có một lệnh init. Điều này sẽ giúp chúng tôi khởi tạo một nhóm node duy nhất. Chúng tôi cũng có thể chọn tham gia một nhóm khác, nhưng các công nghệ bên trong **Docker** Desktop cho Mac và Windows, cách chúng kết hợp với hệ điều hành của bạn để cung cấp trải nghiệm minh bạch, không phải lúc nào cũng giúp kết nối mạng với các node và biểu mẫu **Docker** khác dễ dàng như vậy. một cụm nhiều node . Tôi khuyên bạn nên gắn bó với một node duy nhất và trong trường hợp đó, chúng tôi cần chạy nhóm **docker** và sau đó init để tạo cụm của chúng tôi. Và chỉ trong tích tắc, chúng ta đã hoàn thành. Đó là tất cả những gì cần thiết để lật công tắc đó. Câu hỏi. Bạn nghĩ điều gì sẽ hiển thị bây giờ nếu tôi chạy lại lệnh thông tin **docker**? Bây giờ khi chúng ta cuộn lên đây, chúng ta có rất nhiều thông tin bên dưới phần **Swarm**. Trong trường hợp này, một node đơn lẻ tụ lại với một node người quản lý duy nhất. Và đáng chú ý nhất, chúng tôi có hoạt động thay vì không hoạt động, cho chúng tôi biết rằng công tắc đang được bật.

### **Cloning the Course Repo and Explaining the Multi-machine Vagrantfile**

Được rồi, đối với ví dụ tiếp theo của chúng ta, tôi muốn xem xét thiết lập nhiều node bằng cách sử dụng Vagrant để tạo máy ảo cục bộ. Và để làm được điều đó tôi đã chuẩn bị sẵn một số tệp cho việc này trong kho khóa học này, sẽ có một liên kết trong các tệp bài tập. Lấy một bản sao của kho lưu trữ này và sau đó chuyển đến bất cứ nơi nào bạn muốn sao chép nó. Ví dụ, tôi sẽ đặt nó vào **Swarm** - gs ngay trong thư mục chính của mình. Giữ cho việc vỗ nhẹ đơn giản. Thay đổi vào thư mục đó và liệt kê các tệp. Và bên trong có một thư mục phòng thí nghiệm. Bên trong thư mục labs là thư mục người lang thang. Và bên trong đó, bạn sẽ tìm thấy các tệp khác nhau cho ví dụ này, đặc biệt là Vagrantfile. Chúng tôi sẽ mở nó bằng Visual Studio Code và chỉ cần xem nhanh thư mục labs trong Vagrantfile đó. Bạn không cần phải là một chuyên gia tại Vagrant. Trong thực tế, nếu bạn không, tôi đã thiết lập README này với hướng dẫn cho các yêu cầu trước, cách cài đặt Vagrant, tôi đề xuất VirtualBox. Nếu bạn đã quen với một hypervisor khác, điều đó là tốt. Bạn có thể cắm nó với tùy chọn nhà cung cấp hoặc sử dụng biến môi trường VAGRANT\_PROVIDER. Ở bên dưới, tôi có danh sách các lệnh khác nhau mà tôi nghĩ có thể có lợi, và nó khá dài, nhưng không sao, tôi sẽ xem qua các lệnh này. Tôi chỉ muốn bạn có những thứ này để tham khảo cho việc sao chép / dán và cũng để tra cứu mọi thứ. Nếu tôi không đề cập đến điều gì đó, tôi đã cố gắng trình bày ở đây cho bạn những điều phổ biến mà bạn có thể muốn làm. Sau đó, những gì bạn có với Vagrant là một Vagrantfile xác định máy ảo của bạn, giống như một **docker** - tệp soạn thảo xác định một ứng dụng và tất cả các dịch vụ liên quan, khối lượng và mạng, điều này cũng xác định các máy ảo mà bạn muốn quay lên . Vì vậy, quay trở lại sự phân đôi của chúng tôi. Chúng tôi đang tập trung vào lớp node ngay bây giờ. Được rồi, vì vậy bên trong Vagrantfile này, tôi sẽ cho bạn hiểu sâu hơn về nó bởi vì tôi đang sử dụng các vòng lặp ở đây bên trong cú pháp Ruby để cấu hình, một DSL được xây dựng bên trong Ruby. Tôi có hai vòng lặp lần lượt trên ba và bốn số để xác định tối đa ba node người quản lý và bốn node công nhân. Bây giờ điều đó không có nghĩa là bạn phải xoay tất cả những thứ này cùng một lúc, nhưng chúng ở đó nếu bạn muốn và bạn có thể thay đổi phạm vi nếu bạn muốn mở rộng hoặc giảm những phạm vi đó. Bên trong đây, chúng tôi đang xác định các máy ảo. Tôi đặt tên cho chúng là m và sau đó là bất kỳ số nào trong vòng lặp, vì vậy m1, m2, m3, và sau đó cho mỗi cái tôi đang lấy node đó đại diện cho VM. Về cấu hình cho Vagrant, tôi chỉ định cho mạng và tên máy chủ các giá trị mà tôi muốn. Tôi muốn đặt tên máy chủ là m và sau đó là 1, m2, v.v., để dễ tham khảo. Và sau đó tôi cũng muốn thiết lập một mạng riêng và đặt địa chỉ IP một cách rõ ràng chỉ để làm cho cuộc sống dễ dàng hơn một chút khi liên kết các node lại với nhau. Vì vậy, tôi đang sử dụng 99,20 và sau đó là 1, 2, 3 cho người quản lý và 21 và sau đó là 1, 2, 3, 4 cho người lao động. Các công nhân đều được đặt tên w và 1, w và 2, v.v., và tên máy chủ cũng tương tự như vậy. Ở bên dưới, một điều bạn có thể muốn thay đổi là cài đặt của bạn cho bộ nhớ VirtualBox và phân bổ CPU của bạn. Tùy thuộc vào máy của bạn và những gì bạn có sẵn cho tài nguyên, bạn có thể không muốn phân bổ quá nhiều cho mỗi máy. Bạn cũng không phải xoay tất cả những thứ này cùng một lúc, nhưng nếu bạn muốn điều chỉnh các cài đặt này, bạn có thể. Và sau đó cho phần còn lại của điều này, tôi đang sử dụng trình điều khiển **Docker**. Điều này, đằng sau hậu trường, sử dụng get.**docker**.com nếu bạn đã từng sử dụng tập lệnh tiện lợi đó. Vì vậy, Vagrant đang sử dụng điều này, cũng như một số chỉnh sửa khác để cài đặt Vagrant vào máy ảo cho bạn bằng cách sử dụng những gì nó đề cập đến như một người cung cấp **Docker**. Và tôi không có đối số nào, vì vậy tất cả những gì tôi đang làm là cung cấp để đảm bảo rằng **Docker** đã được cài đặt. Bạn có thể cài đặt theo cách khác nếu muốn. Ví dụ, tôi theo dõi điều này với một trình điều chỉnh cho một tập lệnh shell. Nhà cung cấp chạy sau khi máy ảo được tạo lần đầu tiên. Trong trường hợp này, tôi đang cài đặt thêm một số phần mềm, công cụ mà tôi thích cho mục đích gỡ lỗi. Danh sách này có thể mở rộng. Bạn cũng có thể thêm của riêng bạn. Chỉ là những thứ tôi thích bên trong VM khi tôi vào đó và thăm dò xung quanh để xem chuyện gì đang xảy ra. Và tôi không thể giới thiệu kịch bản thuận tiện đó đủ. Tôi nghĩ rằng đó là một cách tuyệt vời khi nói đến việc học chỉ để nhanh chóng tạo ra các môi trường. Nhân tiện, tôi có một khóa học về Quản lý **Docker** trên Máy chủ Linux. Bạn có thể truy cập trang đó nếu bạn muốn biết bất kỳ điều gì về quá trình cài đặt. Tôi thậm chí còn bao gồm các kịch bản thuận tiện. Nhân tiện, chúng chỉ được đề xuất cho mục đích thử nghiệm hoặc để học tập, không phải cho môi trường sản xuất. Và sau đó ở bên dưới, tôi có một số ghi chú bổ sung. Ví dụ: liên kết đến tài liệu cho Vagrant, liên kết đến một công cụ để tìm kiếm các hộp. Hộp là những gì tôi đã bỏ qua ở trên cùng ở đây. Các hộp xác định một image, nếu bạn muốn, đối với máy ảo, giống như một image cho một vùng chứa. Trong trường hợp này, tôi đang sử dụng Ubuntu 21 - 04. Dưới đây là một số bản dựng khác từ Vagrant Cloud mà bạn có thể chọn hoặc thậm chí bạn có thể tìm kiếm một bản phân phối hoặc hệ điều hành khác, có lẽ là CentOS thay vì Ubuntu.

Vagrant Up m1 w1 - Tự động cung cấp hai máy ảo Ubuntu với **Docker** Engine Running

Và điều thực sự gọn gàng là tất cả những gì tôi phải làm là chuyển sang dòng lệnh, ngay nơi tôi đã kiểm tra các tệp này. Tôi có Vagrantfile đó ở đây, hãy đảm bảo rằng nó nằm trong cùng một thư mục mà bạn đang ở trong đó và chỉ cần chạy lệnh vagrant up và chỉ định những gì bạn muốn up. Có lẽ tôi sẽ giới thiệu một nhân viên và một node quản lý. Được rồi, tôi sẽ loại bỏ điều đó và có thể trong vài giây hoặc vài phút, hộp sẽ được kéo xuống nếu nó chưa được mở và sau đó Vagrant sẽ trải qua quá trình cung cấp các máy ảo để bạn có thể truy cập chúng qua SSH thực sự dễ dàng. Được rồi, chúng ta bắt đầu. Bạn có thể thấy một số gói đang cài đặt. Ở phía bên tay trái , Vagrant mở ra hộp mà nó đang hoạt động, vì vậy m1, w1, bạn có thể xem tiến trình của từng người trong số họ và nếu bạn tò mò, bạn có thể mở VirtualBox GUI và bạn sẽ thấy thông tin được phản ánh ở đây đối với các hộp mà Vagrant đã tạo, vì vậy m1, w1, có tiền tố là tên dự án hoặc tên thư mục của Vagrantfile, ở đây. Có thể thấy chúng ta đang cài đặt **Docker** trên máy. Được rồi, các lệnh đã hoàn thành ở đó. Nếu tôi muốn kiểm tra trạng thái, tôi có thể làm điều đó. Bạn có thể thấy m1 đang chạy và w1 đang chạy.

Sử dụng Cờ --advertise-addr với **Docker** **Swarm** Init trên các máy có nhiều NIC

Chúng tôi đã có node macOS của mình mà chúng tôi đã biến thành một nhóm node duy nhất và bây giờ chúng tôi đã thực hiện quá trình xoay vòng hai node với Vagrant để biến chúng thành một nhóm. Đó sẽ là một quá trình gồm hai bước khởi tạo một trong số chúng dưới dạng một node bầy đàn và sau đó kết hợp node còn lại với nó. Được rồi, chúng ta có một số cách để giải quyết vấn đề này. Đầu tiên, chúng ta có thể ssh vào những máy này. Vì vậy, chúng ta sẽ đi vào m1 trước. M1 là viết tắt của manager1. Chúng ta sẽ nói về vai trò của node trong giây lát. Hãy kết thúc điều này ở đây. Và bây giờ chúng ta hãy tiếp tục và chăm sóc tạo ra bầy đàn của chúng ta. Vì vậy, trước tiên, tôi muốn xem trạng thái của chúng tôi là gì, vì vậy thông tin về **docker** sẽ cho tôi biết điều đó. Nếu tôi cuộn lên đây, bạn có thể thấy, được rồi, chúng tôi không hoạt động tại thời điểm này. Thật tuyệt. Xóa nó ra. Bạn có nhớ lệnh tôi đã sử dụng để tạo bầy đàn là gì không? Vì vậy, chúng ta cần gõ **docker** và sau đó **Swarm**. Nếu bạn cần trợ giúp, chỉ cần nhớ rằng bạn có thể nhấn Return mà không cần bất kỳ đối số nào và nó sẽ hiển thị cho bạn nhiều lệnh con khác nhau và chúng tôi muốn lệnh con init. Được rồi, hãy tắt nó đi, và ồ - không, chúng ta có một vấn đề ở đây. Vì vậy, điều này xuất hiện khi chúng ta có nhiều địa chỉ IP được gán cho một thiết bị. Trong trường hợp này, nó thực sự hiển thị cho chúng ta hai địa chỉ IP đang gây ra sự cố hoặc chúng ta có thể phát hiện ra điều đó bằng cách chạy lệnh để xem các giao diện khác nhau trên thiết bị của mình. Bạn có thể xem 99.201 và 10.0.2.15. Vì vậy, chúng ta chỉ cần thông báo cho daemon cái nào được sử dụng để quảng cáo làm cổng giao tiếp giữa các node trong cụm. Ở đó chúng tôi có hai sự lựa chọn. Và nếu bạn tò mò tại sao, tôi đã tiếp tục và mở VirtualBox hypervisor của mình tại đây, tôi đã chọn m1. Tôi sẽ đi đến Cài đặt, chuyển đến tab Mạng, nó cung cấp cho tôi một chút chi tiết hơn những gì tôi có thể nhận được trên màn hình tóm tắt phía trước cũng có một số thông tin này, nhưng bạn có thể thấy chúng tôi có một bộ điều hợp được thiết lập như một bộ điều hợp NAT. Đó là giao diện mà Vagrant đang kết nối. Và sau đó chúng tôi có Bộ điều hợp 2, Bộ điều hợp chỉ dành cho máy chủ lưu trữ của chúng tôi , vì vậy đó là mạng riêng của chúng tôi trong VirtualBox. Đó là những gì chúng tôi đã định cấu hình với địa chỉ 99,201. Nếu bạn muốn xác nhận đó là địa chỉ nào, hãy vào mục Nâng cao ở đây và có sẵn địa chỉ MAC, 0800 là một trong số đó, sau đó là địa chỉ còn lại, 0263 cho 10.0.2.15. Vì vậy, điều đó không phù hợp. Nếu tôi xem qua Bộ điều hợp 1, bạn có thể thấy điều này được mở rộng ngay bây giờ, 026384, khớp với địa chỉ MAC cho 10.0.2.15, là giao diện NAT mà chúng tôi không muốn sử dụng. Chúng tôi muốn sử dụng địa chỉ IP riêng, sẽ là Bộ điều hợp 2. Vì vậy, đó là cách bạn có thể xác nhận sử dụng cái nào dựa trên chiến lược mạng và thiết lập cho máy ảo cụ thể của bạn trong trường hợp này. Hoặc nếu đó là một máy vật lý, nếu bạn có nhiều NIC, đó sẽ là giao diện bạn muốn sử dụng để nói chuyện với các máy vật lý khác của mình vì vào cuối ngày, các node của chúng ta phải có thể nói chuyện với nhau về một điểm chung mạng. Vì vậy, bây giờ chúng ta biết chúng ta muốn sử dụng cái nào, chúng ta có thể chỉ cần sao chép địa chỉ IP. Và tôi có thể lên đây và chạy lại lệnh của mình. Tôi muốn chỉ định địa chỉ được quảng cáo và chỉ cần dán địa chỉ đó vào đây, số 99.201. Và bạn hiểu rồi, bây giờ chúng tôi đã khởi tạo cụm của chúng tôi. Đố nhanh. Tôi có thể làm gì ở đây để kiểm tra trạng thái của node cụ thể này về việc liệu chế độ bầy đàn đó có được bật hay không? Chà, đó là một cuộc gọi đơn giản đến thông tin **docker** và tôi sẽ nhận được đầu ra mở rộng và tôi thực sự đã có đầu ra mở rộng trong phần **Swarm**. Bầy đàn hiện đang hoạt động. Chú ý rằng nó cũng là một người quản lý. Node đầu tiên tham gia sẽ là một người quản lý, nếu không thì cụm của bạn không thể hoạt động. Và sau đó có các cài đặt khác nhau được mặc định ở đây. Và nếu tôi chia màn hình ở đây, tôi sẽ quay lại trong kết nối thứ hai với máy ảo của tôi. Tôi sẽ chạy lại **docker** **Swarm** init. Lần này tôi chỉ cần giúp đỡ cho việc này. Nếu bạn so sánh và đối chiếu phần trợ giúp init và các cài đặt khác nhau ở đây, bạn sẽ bắt đầu thấy một số chồng chéo. Ví dụ, cần có một cài đặt ở đây cho kích thước lưu giữ nhiệm vụ, nhiệm vụ - lịch sử - giới hạn, chúng ta bắt đầu. Nó được mặc định là 5, và bạn có thể thấy nó được đặt mặc định là 5 ngay tại đây. Tôi cũng có Nhóm Địa chỉ Mặc định. Hãy đến đây, default - addr - pool, đó là cách chúng ta có thể kiểm soát điều đó từ dòng lệnh, cũng như SubnetSize ở đây. Chúng tôi cũng có Cổng đường dẫn dữ liệu. Điều đó phù hợp với. Vì vậy, rất nhiều cờ mà chúng tôi có thể chuyển đến **docker** **Swarm** init sẽ chỉ định đầu ra của lệnh thông tin của chúng tôi ở đây và thực sự ra lệnh cho các tham số cấp độ cụm mà chúng tôi đang làm việc. Các thông số này không phải tất cả đều được thiết lập trong đá. Ví dụ: bạn có thể sử dụng lệnh cập nhật nhóm **docker** để cập nhật một số trong số chúng. Ví dụ, đây là nhiệm vụ - lịch sử - giới hạn, hoặc lưu giữ. Và sau đó những người khác được cập nhật bởi các khía cạnh khác của cụm, ví dụ như mạng. Và nhiều trong số này là các cấu hình nâng cao mà bạn sẽ quan tâm khi đến lúc triển khai phiên bản sản xuất. Bây giờ, bắt đầu, đừng lo lắng quá nhiều về những điều này, chỉ cần biết rằng chúng ở đó khi thời gian đến.

Quản lý node nhanh hơn với quyền truy cập SSH trực tiếp - Tận dụng ssh-config của Vagrant

Được rồi, vì vậy tôi đã mở 2 kết nối với VM m1 của mình và tôi gợi ý rằng đây là một cách để kết nối với VM và kiểm soát quá trình khởi tạo của bầy đàn của chúng tôi để tương tác với daemon **Docker**. Và trong khi điều này hoạt động, có một chút chi phí liên quan đến việc chuyển đổi giữa các máy ảo, hoặc có thể mở một cửa sổ thứ hai như chúng ta phải làm. Vì vậy, những gì tôi muốn làm là rời khỏi đây, trở lại máy chủ và tôi sẽ đóng một trong những cửa sổ này. Nếu bạn nhìn vào các tệp ở đây, có một thư mục máy chủ lưu trữ và bạn không cần phải chạy các tập lệnh này để làm theo. Tôi thiết lập những thứ này chủ yếu cho mục đích của tôi. Nhưng những tập lệnh này thực hiện một số điều. Điều quan tâm đầu tiên là máy chủ lưu trữ - xác minh. Tất cả những gì tôi làm chỉ là kiểm tra cấu hình của những gì tôi sắp đặt. Không có thiết lập cấu hình ssh. Và về ngữ cảnh của **docker**, đó là những gì chúng tôi sẽ sử dụng ở đây để kết nối với máy ảo từ xa của chúng tôi, bạn có thể thấy chúng tôi chỉ có mặc định. Vì vậy, đó là một bức ảnh chụp trước. Bây giờ, những gì tôi muốn làm chỉ là chạy tập lệnh thiết lập. Điều này sẽ tìm thấy các máy ảo đang chạy, trong trường hợp của chúng tôi, đó phải là m1 và w1. Nó sẽ tiếp tục và kết xuất cấu hình SSH cho những người từ vagrant bằng cách sử dụng lệnh vagrant ssh - config. Và điều đó cần thiết vì hai lý do. Nó nhanh hơn chỉ để sử dụng chính ssh. Vagrant có một số chi phí để kết nối qua ssh. Nó thuận tiện, nhưng nó có chi phí cao, vì vậy đó là lý do tại sao lệnh này tồn tại. Cũng cần thiết khi sử dụng tính năng ngữ cảnh **docker**, cụ thể là ngữ cảnh SSH. Chúng tôi muốn có cấu hình của chúng tôi cho kết nối ssh đó được nhồi vào tệp cấu hình SSH để chúng tôi có thể chỉ cần trỏ vào đúng máy chủ và thực hiện với nó về mặt cấu hình ngữ cảnh. Và tôi sẽ cho bạn thấy nó trông như thế nào bởi vì đó là những gì chúng tôi đã làm ở đây. Chúng tôi đã tạo ra hai bối cảnh riêng biệt, một cho m1 và một cho w1. Và nếu chúng ta có nhiều máy ảo đang chạy hơn, tập lệnh này sẽ tạo ra nhiều hơn. Nếu tôi chạy lại tập lệnh xác minh của mình, thì trước tiên, bạn có thể thấy tệp tồn tại mà tôi muốn tạo ở đây. Điều này liên quan đến cấu hình SSH của tôi. Tôi chỉ vào thư mục config.d này và bao gồm tất cả các cấu hình trong đó. Vì vậy, những gì tôi đã làm chỉ là đưa đầu ra từ cấu hình vagrant ssh vào thư mục này để cuối ngày, tôi có thể gõ lệnh ssh trong m1 và giống như với vagrant ssh và m1. Bây giờ nó chậm hơn một chút với người lang thang.

Quản lý node minh bạch với các khung cảnh **Docker** dựa trên SSH

Vì vậy, đó là những gì mà phần đầu tiên làm. Phần thứ hai ở đây để liệt kê ngữ cảnh **docker**, tôi tạo hai ngữ cảnh, m1 và w1, và tôi sử dụng ssh cho ngữ cảnh đó, và chỉ cần trỏ vào máy chủ m1, và tôi có thể làm được điều đó nhờ cách Vagrant viết ra cấu hình ssh. Hãy để tôi lấy ra tệp đó cho bạn. Vì vậy, đây là tệp mà chúng tôi đã tạo. Tất cả những gì chúng tôi đang làm ở đây là đặt tên cho máy chủ, cung cấp cổng IP của máy chủ, trong trường hợp này là sử dụng giao diện khác mà Vagrant đã thiết lập. Đó là lý do tại sao tôi sử dụng Vagrant để tạo điều này bởi vì những số cổng này có thể thay đổi khi chúng tôi thêm nhiều máy hơn, chẳng hạn như chúng sẽ không giống nhau. Nó cũng chỉ định đúng người dùng và sau đó quan trọng nhất là tệp danh tính mà Vagrant tạo ra duy nhất cho máy ảo cụ thể này, nó tạo ra một kết nối an toàn. Bạn sẽ nhận thấy nó nằm trong cùng một thư mục mà chúng tôi đang làm việc bên trong. Có một thư mục .vagrant. Vì vậy, ngay đây là repo chúng tôi đã kiểm tra và nằm trong thư mục labs / vagrants. Và bên trong đó, có một thư mục .vagrant và bên trong đó, được lồng vào sâu bên trong là một khóa riêng cho mỗi máy. Vì vậy, có w1 và có m1. Vì vậy, chúng tôi chỉ cần chuyển tất cả cấu hình này vào ssh, vậy là nó đã sẵn sàng hoạt động. Vì vậy, khi chúng tôi thiết lập ngữ cảnh **docker** của mình, chúng tôi chỉ cần chuyển URL với tên máy chủ trong đó. Mọi thứ khác được lấy từ tệp cấu hình đó. Và điều thú vị là bạn có thể thấy dấu hoa thị bên cạnh mặc định là ngữ cảnh hiện tại, chúng ta có thể thay đổi điều đó. Có một lệnh sử dụng và chúng ta có thể nói sử dụng m1 hoặc sử dụng w1. Và tất nhiên nếu tôi liệt kê ra bối cảnh, bạn nghĩ điều gì sẽ xảy ra bây giờ? Chà sẽ không có gì thay đổi, và đó là bởi vì tôi vẫn ở trên máy chủ của mình về mặt khách hàng, và vì vậy cấu hình máy khách là những gì chúng tôi đang xử lý theo ngữ cảnh và vì vậy điều đó sẽ không thay đổi. Bây giờ, phần tiếp theo, giả sử tôi muốn nhận một số thông tin về công tắc **Swarm** đó để xem nó có được kích hoạt hay không, tôi có thể nhập gì vào đây?

**Docker** Node ls + Chuyển đổi ngữ cảnh **Docker** liền mạch

Được rồi. Vì vậy, đó là lệnh thông tin **docker**. Và sau đó từ một kết nối từ xa qua SSH, tôi có thể chạy một lệnh như thể tôi đang SSHd vào máy ảo cụ thể đó, ngoại trừ tôi không phải làm điều đó. Tôi đang ngồi ngay đây trên máy chủ của tôi. Nếu tôi cần cửa sổ thứ hai, tôi có thể bật mở cửa sổ thứ hai, có thể liệt kê ra ngữ cảnh của mình. Bạn có thể thấy tôi vẫn đang sử dụng m1 làm bối cảnh hiện tại, điều đó khá tuyệt và tôi có thể thực hiện một số lệnh khác như **docker** node ls, một cách khác để xem các node trong một cụm. Và vì vậy, tôi hy vọng bạn có thể thấy đây là một cơ chế mà chúng ta có thể tương tác từ xa với các node khác. Và trong bối cảnh nhóm và cụm, chúng tôi muốn làm điều đó cho mục đích quản lý cụm, quản lý các node, vòng đời của chính node đó, không nhất thiết để triển khai bất kỳ loại phần mềm nào, đó là vai trò của chính nhóm. Vì vậy, chỉ để chứng minh tôi được kết nối với một node khác, tôi sẽ sử dụng lại kết nối mặc định và tôi sẽ chạy node **docker** ls, điều này sẽ xảy ra trên máy Mac của tôi, mà chúng tôi đã thiết lập vào ngày hôm trước với một nhóm. Và bạn có thể thấy chúng tôi có một tên máy chủ khác ở đây, một ID khác và cả hai đều là những nhà lãnh đạo, điều này sẽ không xảy ra trong cùng một nhóm bầy đàn, vì vậy đây là tất cả bằng chứng rằng chúng tôi chỉ đang chuyển đổi giữa hai kết nối khác nhau nơi đây. Trên thực tế, tôi có thể tiếp tục và sử dụng w1 để thay thế, và nếu tôi nhập vào node **docker** ls, bạn nghĩ điều gì sẽ hiển thị ở đây? Vì vậy, chúng tôi đang ở trên w1 ngay bây giờ, nó không phải là một phần của một bầy và vì vậy chúng tôi nhận được một thông báo lỗi cho biết: Này, lệnh này là một lệnh quản lý bầy, do đó, node này không phải là một phần của một bầy , không có tiện ích khi chạy lệnh này. Nó nói rằng chúng ta có thể chạy một **docker** **Swarm** init để tạo một cluster hoặc chúng ta có thể chạy một **docker** **Swarm** tham gia nếu chúng ta muốn kết nối với một cái hiện có, vì vậy hãy làm điều đó trong video tiếp theo.

Trực quan hóa các khung cảnh **Docker** để đơn giản hóa việc quản lý nhiều node

Vì vậy, đây là cấu trúc liên kết hiện tại của chúng tôi. Và để thực hiện bước tiếp theo để tham gia từ w1 đến m1, tôi sẽ tận dụng ngữ cảnh để cho bạn thấy nó có thể thực sự đơn giản như thế nào mà không cần SSH vào VM. Tôi nghĩ rằng ngữ cảnh là một phần rất quan trọng trong việc quản lý một cụm. Khi bạn cần vào và ra khỏi các node thực sự nhanh chóng, đó có lẽ là cách tốt nhất để làm điều đó. Vì vậy, hãy nhớ tất cả những điều này ngay bây giờ là trên một máy, máy Mac của tôi được cài đặt **Docker** Desktop, có một nhóm node duy nhất và sau đó chúng tôi có hai máy ảo ảo của chúng tôi, một trong số đó cũng được khởi tạo dưới dạng một nhóm node duy nhất. Và bây giờ ở phía bên trái đây , tôi sẽ kéo lên những gì trông giống như một lời nhắc và bắt đầu với một chút mã mà chúng tôi đang làm việc, màu - được mã hóa để giúp hiểu ngữ cảnh. Những thứ này sẽ rất quan trọng trong tương lai vì tôi nghĩ chúng là một công cụ tuyệt vời để quản lý các node trong một cụm. Ngay cả sau khi bạn đã liên kết các node với nhau, điều này vẫn có giá trị để vào và ra thực sự nhanh chóng mà không cần phải SSH đầy đủ vào một máy. Thực ra nó gần giống như một kết nối SSH trong suốt. Được rồi, ở trên cùng ở đây, chúng ta có cấu hình vagrant ssh của mình. Hãy nhớ rằng chỉ cần viết ra cấu hình để truy cập các máy ảo ảo, đặt nó ở vị trí thuận tiện và đặt cho mỗi máy ảo của chúng ta một cái tên thân thiện là m1 và w1. Bằng cách đó, chúng tôi không phải tham chiếu đến họ bằng địa chỉ IP, chúng tôi không phải cung cấp tên người dùng, chúng tôi không phải tìm cách cung cấp tệp khóa riêng tư để truy cập chúng, v.v. Tất cả được gói gọn trong tệp cấu hình đó cho chúng tôi. Và sau đó, hai dòng tiếp theo sẽ trông như thế nào nếu bạn chạy lệnh tạo ngữ cảnh thay vì sử dụng các tập lệnh mà tôi đã có và nếu bạn gặp bất kỳ sự cố nào với các tập lệnh, bạn có thể sử dụng mã mẫu này ngay tại đây. Nó sẽ được kiểm tra trong repo khóa học; một lý do khác tại sao tôi muốn hiển thị điều này ở đây. Vì vậy, chúng tôi sẽ tạo hai ngữ cảnh, một ngữ cảnh màu xanh lam, một màu xanh lục và một ngữ cảnh màu lục lam, sau đó tôi quay lại và liệt kê ra ngữ cảnh **Docker**. Đầu tiên là bối cảnh màu đỏ hoặc hơi hồng của chúng ta. Điều đó chỉ ra nhóm node đơn **Docker** Desktop ngay bây giờ. Đó chỉ là mặc định được mã hóa cứng. Bạn không thể thay đổi điều đó. Và tiếp theo chúng ta có m1. Về cơ bản, chúng tôi đang đặt bí danh cho giao thức SSH trỏ vào tên thân thiện đó của m1, đó là ssh: // m1, chúng tôi đang đặt bí danh đó với m1 cho một ngữ cảnh trong **Docker** và sau đó chúng tôi làm điều tương tự đối với w1, ngữ cảnh màu lục lam . Được rồi, sau đó chúng tôi sử dụng các ngữ cảnh này và tôi đã nắm bắt được bản chất của những gì chúng tôi đã làm với chúng ở đây bằng một số lệnh được sửa đổi, nhưng chúng sẽ có ý nghĩa trong ngữ cảnh mà chúng tôi vừa chạy. Vì vậy, ngữ cảnh đầu tiên, ngữ cảnh mặc định, là ngữ cảnh được đặt theo mặc định cho đến khi chúng ta thay đổi điều gì đó. Vì vậy, tôi đang sử dụng lệnh hiển thị ngữ cảnh **docker** và bạn có thể thấy nó là mặc định. Bất cứ khi nào bạn chạy lệnh hiển thị đó, bạn sẽ thấy bất cứ điều gì mà bạn đã chọn cho ngữ cảnh hiện tại. Vì mặc định là ngữ cảnh hiện tại của chúng tôi, bạn có thể thấy chúng khi tôi chạy thông tin **docker** xuống bên dưới và lấy một số trường ra khỏi kết quả. Chúng tôi đang chỉ vào một nhóm node duy nhất của chúng tôi, vì vậy chúng tôi có một nhóm đang hoạt động. Bạn có thể thấy địa chỉ IP. Chúng ta có thể thấy chỉ có một node trong cụm. Và chúng ta có thể thấy tên là **docker** - máy tính để bàn xác nhận rằng ngữ cảnh của chúng ta đã chỉ ra vị trí mà chúng ta nghĩ. Bây giờ nếu tôi muốn kết nối với m1 thay thế, tôi phải làm gì khác với lệnh tiếp theo? Chà, hãy xem phần tiếp theo của kịch bản ở đây. Vì vậy, chúng ta chỉ cần chạy một use m1 và chúng ta sẽ chuyển chúng ta sang ngữ cảnh m1 và trỏ chúng ta đến node m1 đó. Chạy lệnh chính xác như trên, bạn có thể thấy chúng tôi cũng có một nhóm đang hoạt động vì đó là nhóm node đơn của chúng tôi. Có địa chỉ IP và đó sẽ trông quen thuộc như địa chỉ IP của node quản lý đầu tiên nếu bạn nhìn vào tệp vagrant và sau đó bạn có thể xác nhận tên của máy là m1. Và tôi có một lệnh cuối cùng ở đây. Điều này sẽ khác một chút so với những gì chúng tôi đã làm ở trên. Hãy để tôi chỉ cho bạn điều này ở đây. Vì vậy, tôi chỉ định ngữ cảnh bằng một đối số dòng lệnh ở đây cho **Docker** CLI. Do đó **docker**, và sau đó a - c w1 chọn ngữ cảnh w1, ngữ cảnh này sẽ hướng chúng ta đến node w1. Phần còn lại của lệnh này giống như trước, vì vậy bạn có thể thấy chúng tôi có một nhóm không hoạt động và tên của node là w1, hãy xác nhận rằng chúng tôi đã chuyển đổi ngữ cảnh. Bây giờ, tôi có một câu hỏi cho bạn. Nếu tôi chạy một lệnh **Docker** khác sau đó và không chỉ định ngữ cảnh tại dòng lệnh nữa, thì tôi sẽ tương tác với ngữ cảnh nào tiếp theo? Vì vậy, hãy tưởng tượng bỏ đối số - c đó xuống lệnh cuối cùng và chạy lại nó. Vì vậy, khi tôi chỉ định ngữ cảnh tại dòng lệnh làm đối số cho **Docker** CLI, lúc đó tôi chưa chuyển đổi ngữ cảnh và do đó các lệnh tiếp theo sẽ thực sự kích hoạt đối với m1. Đó là ngữ cảnh cuối cùng mà chúng ta chuyển sang sử dụng lệnh con. Điều thú vị về điều này là nếu bạn có hai hoặc ba máy chủ mà bạn cần thực hiện các lệnh để thực sự nhanh chóng, bạn có thể chỉ cần lặp lại chúng và rút ra lệnh, thêm một đối số vào nó và bạn có thể thực hiện các lệnh **Docker** của mình. Đừng lo lắng về việc cuối cùng bạn sử dụng ngữ cảnh nào, nó sẽ là bất cứ thứ gì bạn bắt đầu bởi vì bạn đã không chuyển đổi bất kỳ ngữ cảnh nào trong suốt quá trình đó. Có lẽ bạn có thể nhớ lại biến môi trường **docker**\_host và họ liên kết của nó với các biến khác để định cấu hình cách bạn truy cập vào một node. Và có lẽ bạn thậm chí có thể nhớ trước khi chúng ta có khả năng sử dụng SSH như một giao thức để truy cập vào node. Bạn đang sử dụng TCP hoặc có thể đang cố chạy một ổ cắm TCP an toàn. Nếu bạn đã làm điều đó, bạn biết có khá nhiều đối số phải được thông qua. Chúng vẫn ở đó, chúng chỉ là tất cả ẩn sau bối cảnh cho bạn. Điều đó nói rằng, tôi không thể khuyên bạn nên sử dụng đủ ngữ cảnh SSH bất cứ khi nào có thể. Và tôi nói rằng vì A, nó an toàn và B, nó dễ dàng cấu hình hơn rất nhiều, vì bạn có thể thấy các lệnh ở trên giải quyết vấn đề đó cho chúng ta. Chỉ cần định cấu hình SSH, và nếu chúng ta đã có, thì chúng ta không cần phải làm gì khác hơn là sử dụng SSH.

Tham gia các node một cách chính xác với các lệnh một lần qua **Docker** Contexts

Hãy áp dụng những gì chúng ta đã học về ngữ cảnh để lấy một nhóm node của chúng ta và biến nó thành nhóm hai node của chúng ta. Được rồi, vì vậy tôi đang ngồi lại đây trên máy chủ của mình và tôi muốn thể hiện một chút thay đổi mà tôi đã thực hiện đối với môi trường của mình. Trước hết, tôi đã cắt bớt lời nhắc của mình để chỉ hiển thị thư mục hiện tại mà tôi đang ở trong đó. Vì vậy, tôi vẫn đang ngồi bên cạnh tệp vagrant của chúng tôi chỉ để cho chúng tôi thêm dung lượng vì tôi muốn thêm một tính năng và tôi đã tạo một node chuyển đổi cho điều này để nó bật tính năng này, điều đó sẽ hiển thị cho tôi ngữ cảnh **Docker** mà tôi đã chọn , trong trường hợp này, m1. Vì vậy, nếu tôi sử dụng một cái khác, bạn có thể thấy nó chuyển sang mặc định. Và, tất nhiên, tôi có thể bật và tắt tính năng đó khi tôi thấy phù hợp. Vì vậy, nếu bạn thấy điều đó trong suốt khóa học này, bạn sẽ biết đó là gì. Về cơ bản, đó là một móc nhỏ bên trong trình bao của tôi đọc lệnh kiểm tra ngữ cảnh của **docker**, không chuyển bất kỳ args nào, điều này sẽ chỉ cung cấp cho tôi thông tin chi tiết về ngữ cảnh được chọn hiện tại. Trong trường hợp này, tôi đang trích xuất tên. Và sau đó, nếu có bất kỳ ghi đè nào đối với máy chủ, chẳng hạn, tôi cũng đang lấy thông tin đó, ví dụ, với **docker**\_host. Vâng, tôi đang lấy thông tin từ đó, đó là nguồn gốc của nó. Tôi đang làm điều đó để bạn có thể thấy khi chúng tôi chuyển đổi giữa các máy mà máy chúng tôi thực hiện lệnh bởi vì nếu không máy chủ có thể khó hiểu khi máy chủ biết chính xác bạn đang nhắm mục tiêu gì, đặc biệt nếu bạn bắt đầu cuộn lại trong lệnh của mình lịch sử và xem xét những gì bạn đã thực hiện. Tôi nói tất cả những điều đó bởi vì điều này thực sự sẽ hữu ích cho chúng tôi. Những gì tôi muốn làm là không làm điều tương tự như m1 về cách thiết lập w1, vì vậy hãy lưu ý vagrant ssh w1. Thay vào đó, tôi muốn sử dụng những gì chúng ta vừa nói với ngữ cảnh để nhanh chóng bật chế độ bầy đàn bằng cách kết hợp nó với cụm bầy đàn đơn node m1. Điều đầu tiên tôi cần làm ở đây là gì? Bởi vì tôi đang thực hiện các thay đổi đối với node w1, tôi sẽ chuyển sang ngữ cảnh đó. Rất có thể, tôi có nhiều hơn một lệnh để chạy chống lại điều đó, vì vậy sẽ rất hữu ích nếu có điều đó làm bối cảnh hiện tại của tôi. Đầu tiên, tôi sẽ cần mã thông báo tham gia. Vì vậy, mặc dù tôi mới chuyển sang w1, tôi thực sự cần một số thông tin từ m1. Bây giờ, tôi sẽ không làm điều đó bằng cách chuyển đổi vì đó chỉ là một phần thông tin mà tôi cần. Thay vào đó, tôi sẽ quay lại đây và tôi sẽ đặt đối số ngữ cảnh toàn cục để ghi đè và thay vào đó nói m1. Và sau đó tôi sẽ nói rằng tôi muốn tham gia - mã thông báo. Và trong trường hợp của w1, tôi muốn đây là một công nhân. Vì vậy, tôi sẽ loại bỏ điều này. Điều này sẽ kết nối với m1 và yêu cầu mã thông báo tham gia để trở thành nhân viên, vì vậy đó là một vai trò. Các node được gán cho vai trò công nhân, về cơ bản chúng bị giới hạn ở việc chỉ thực hiện các nhiệm vụ. Và các nhà quản lý, mặt khác, có quyền quyết định ai thực hiện các nhiệm vụ và theo mặc định, các nhà quản lý cũng sẽ thực hiện các nhiệm vụ. do đó, do sức mạnh của một node người quản lý có thể đưa ra quyết định, mã thông báo sẽ khác vì bạn sẽ không muốn bất kỳ ai có quyền truy cập vào mã thông báo đó, trong khi sự tin tưởng cần thiết để thêm nhân viên không phải là điều đáng lo ngại khi thêm một người quản lý. Bây giờ, điều tuyệt vời là tất cả đều được định dạng cho tôi. Và sau đó tôi chỉ có thể chạy lệnh này ở đây. Và rất may, chúng tôi đã chỉ vào w1 cho một ngữ cảnh, vì vậy bạn có thể thấy node hiện đã được tham gia với tư cách là một nhân viên. Để xác nhận điều đó, tôi có thể lấy thông tin hệ thống và cuộn lên đây, và bạn có thể thấy chúng tôi hiện có một node bầy đàn đang hoạt động. Tôi cũng có thể muốn xem xét các node có trong cụm bây giờ và có vẻ hợp lý khi cho rằng tôi chỉ có thể chạy ls node **docker**, nhưng tôi không thể, và đó là bởi vì tôi không phải là người quản lý. Trong trường hợp của tôi, vì tôi có quyền truy cập vào node người quản lý, tôi có thể chỉ cần kết nối và chạy lệnh một lần trên m1 để xem các node. Và ở đó, bạn có thể thấy chúng tôi có cột HOSTNAME với m1 và w1, trong đó m1 là người quản lý lãnh đạo và w1 hoàn toàn không phải là người quản lý. Bây giờ để thực sự xác nhận điều này, trong video tiếp theo, tôi có một phần mềm nhỏ mà chúng tôi có thể chạy để cung cấp cho chúng tôi image đại diện về cụm của chúng tôi. Hãy xem cách chạy tiếp theo.

Image hóa trạng thái cụm với Trình hiển thị bầy đàn **Docker** (**docker**samples / visualizer) Đang chạy trong Vùng chứa độc lập

Hầu hết thời gian, dòng lệnh chỉ tốt cho việc học. Bạn thậm chí có thể liệt kê các node ở đây và phần nào có được image đại diện trực quan về những gì đang xảy ra với cụm của chúng tôi. Tuy nhiên, bạn phải đi sâu vào các khía cạnh khác nhau để thu thập tất cả các phần thông tin. Và bạn có thể làm điều đó với một loạt các lệnh được xâu chuỗi lại với nhau, nhưng tôi thích một công cụ khác được gọi là **Swarm** Visualizer. Điều này từng được gọi là \_\_\_\_\_ Tôi nghĩ bầy đàn - image hóa. Nó hiện nằm trong kho lưu trữ các ví dụ dưới trình hiển thị. Điều này sẽ cung cấp cho bạn, nếu bạn nhìn vào kho Git cho điều này, là một hình dung đẹp về các máy trong cụm của chúng tôi, cũng như cuối cùng, những gì đang chạy trên chúng khi chúng ta chuyển sang phần tiếp theo mô-đun và nói về các dịch vụ. Để giúp bạn bắt đầu với điều này, nếu bạn xem qua các tệp bài tập, tôi đang ngồi bên trong thư mục viz. In ra thư mục repo bên trong. Và bên trong đây, bạn có thể thấy có một vài tệp khác nhau. Tập trung vào viz - containerner.sh. Đó là một tập lệnh có một lệnh gọi đơn giản bên trong nó để khởi động một vùng chứa với **Docker**. Và, cụ thể, điều này cần được chạy trên node trình quản lý trên cụm của bạn, tùy thuộc vào thiết lập của bạn trông như thế nào. Bởi vì giống như lệnh **docker** node ls không hoạt động trên worker node, vì nó không có tính riêng tư đối với thông tin đó, bạn cần quyền truy cập vào các API trên trình quản lý để có thể lấy dữ liệu để hiển thị trực quan đó. Nếu bạn để ý, ngữ cảnh của tôi được đặt thành m1, có nghĩa là tôi đã sẵn sàng. Vì vậy, tôi sẽ xóa màn hình ở đây. Và một trong hai điều, tôi có thể thực thi tệp đó hoặc tôi đã sao chép và dán lệnh ở đây. Những gì chúng tôi đang làm ở đây là tạo một volume theo cách truyền thống. Vì vậy, bạn nên làm quen với điều này, **docker** container chạy, hoặc có lẽ bạn đã quen với việc chạy **docker**. Chúng tôi đang chỉ định ngữ cảnh m1 để an toàn, mặc dù tôi đã sử dụng ngữ cảnh m1, như bạn có thể thấy ở đây. Đây sẽ là một vùng chứa tách rời, vì vậy tôi muốn nó chỉ chạy ở chế độ nền vì nó sẽ là một image trực quan. Và sau đó tôi có cổng mà chúng tôi truy cập vào nó. Tôi chỉ lập bản đồ nó tới 8080 trên máy chủ. Và sau đó nó cần quyền truy cập vào **Docker** socket để có thể lấy thông tin hiển thị trạng thái của cụm. Và tất nhiên, dòng cuối cùng là image của chính nó. Vì vậy, tôi có thể chạy nó. Hãy dành một chút thời gian để kéo image xuống. Và có vẻ như nó đã có sẵn image. Vì vậy, bây giờ tôi có thể truy cập vào một trình duyệt, tải lên cổng 8080 trên node trình quản lý bằng địa chỉ IP ngay bây giờ và bạn có thể thấy chúng tôi có một image trực quan đẹp mắt. Không có gì chạy bên trong nó, điều này có thể hơi kỳ lạ vì chúng ta có một vùng chứa đang chạy, nhưng nó sẽ không hiển thị ở đây vì nó không phải là một phần của các dịch vụ được quản lý bởi **Swarm** . Nó thực sự đang chạy với các khả năng độc lập của **Docker** Engine. Vì vậy, một điều khác mà tôi muốn chỉ ra khi làm điều này là có một lợi ích lớn đối với việc chuyển đổi chế độ bầy đàn và đó là bạn có thể tiếp tục sử dụng **Docker** theo cách bạn đã luôn sử dụng, cũng như cách mới này với các máy phân cụm cùng. Nếu chúng ta xem qua dòng lệnh ở đây, hãy in ra các vùng chứa của tôi, bạn có thể thấy chỉ vào ngữ cảnh m1 mà chúng ta có trình hiển thị của chúng ta đang chạy. Vì vậy, chúng ta sẽ để điều đó ở đó, và bây giờ hãy thực hiện thêm một số thay đổi đối với cụm của chúng ta và xem những gì xảy ra với trình hiển thị đó.

Tham gia Worker 2 với GUI Web thời gian thực và Giám sát thiết bị đầu cuối

Được rồi. Chúng tôi đã có trình hiển thị đẹp mắt của chúng tôi ở bên phải. Tôi đã chia màn hình thành một phần ba và hai phần ba ở đây để thiết bị đầu cuối có nhiều không gian và trên đầu cửa sổ thiết bị đầu cuối, tôi có một chiếc đồng hồ đang chạy ngay bây giờ chỉ in ra danh sách các node có node **docker** ls . Tôi có một bí danh cho điều này. Nếu bạn muốn xem nó trông như thế nào, tôi có thể xem qua chức năng ở đây cho bạn. Phần quan trọng nhất ở đây là tôi đang chạy một node **docker** ls và chuyển ngữ cảnh cùng với nó để đảm bảo rằng tôi đang kết nối với ngữ cảnh chính xác. Nếu bạn muốn, lệnh watch là những gì tôi đang sử dụng, tôi đã cài đặt nó bằng Homebrew. Không - tiêu đề giảm ngày và giờ và tiêu đề giải thích lệnh đang chạy, điều này thật đáng tiếc, nhưng nó tiết kiệm một số dung lượng. Nó cũng có ở đây, - n 0,5 có nghĩa là làm mới mỗi nửa giây và - d có nghĩa là điểm khác biệt nổi bật. Vì vậy, nếu một giá trị thay đổi, chẳng hạn như sẵn sàng trên một trong các node, chẳng hạn, nó sẽ ngoại tuyến, khi văn bản thay đổi thành một cái gì đó giống như có lẽ xuống, bạn sẽ thấy nhấp nháy trong độ tương phản của các pixel xung quanh khu vực đó để vẽ sự chú ý của bạn đến nó. Được rồi, hãy sao lưu cái này. Điều đó sẽ chạy trên đầu, và sau đó xuống dưới cùng là nơi tôi sẽ thực hiện các lệnh và sau đó chúng ta sẽ có hai cách để xem xét điều này và tôi muốn cung cấp cho bạn hương vị về những gì bạn có thể nghĩ về biểu diễn đồ họa hoặc biểu diễn đầu cuối hoặc biểu diễn dạng văn bản. Tôi thực sự thích cả hai điều này, nhưng hãy sử dụng chúng để xem điều gì sẽ xảy ra khi chúng tôi thêm một node khác vào cụm của mình. Bạn có nhớ tôi có thể làm gì ở đây để kích hoạt một node khác không? Chà, chúng tôi vẫn còn một số, vậy tại sao chúng tôi không kích hoạt w2, hãy đợi một chút và tôi sẽ quay lại khi hoàn tất. Được rồi, w2 đang hoạt động. Tôi có thể xem trạng thái mà tôi đã định cấu hình nếu muốn. Tuyệt quá. Và nếu tôi muốn siêng năng, tôi có thể tiếp tục và chạy tập lệnh thiết lập máy chủ của mình để tạo lại cấu hình SSH của tôi và thêm ngữ cảnh **Docker** cho w2, nhưng bạn có thể xem ở trên cùng, điều gì sẽ xảy ra khi tôi có ngữ cảnh. Bạn có thể thấy một đèn flash ở đó với độ tương phản, đó là điều tôi muốn nói - d. Được chứ. Và nếu tôi muốn, tôi thậm chí có thể xem đầu ra của bối cảnh mới và nhìn vào đó, chúng ta đã có ngữ cảnh w2 của mình. Được rồi. Bây giờ, tôi phải làm gì tiếp theo nếu tôi muốn thêm w2 vào cụm của chúng tôi? Trước hết, tôi cần đảm bảo rằng tôi đang ở trong bối cảnh của một trong các node quản lý. Vì vậy, bạn có thể thấy ở đây tại dấu nhắc thiết bị đầu cuối Tôi có m1 vì vậy tôi rất vui khi đến đó. Sau khi kiểm tra điều đó, tôi biết tôi có thể chạy các lệnh để sửa đổi thành viên trong cụm và nếu tôi không nhớ chính xác tham gia - mã thông báo là gì, tôi đã không viết nó ra, ví dụ: tôi chỉ có thể chạy **docker** **Swarm** tham gia - lệnh mã thông báo và sau đó tôi cần cung cấp nhân viên hoặc người quản lý. Và vì vậy trong trường hợp này, chúng ta đang nói về một công nhân và điều đó sẽ làm nảy sinh điều đó. Tôi tin rằng cũng có một chế độ yên tĩnh, vâng, bạn hiểu rồi, chế độ yên tĩnh nơi bạn có thể tự lấy ra mã thông báo nếu đó là tất cả những gì bạn muốn, nhưng tôi thích chế độ dài dòng hơn vì tôi có thể tính toán điều này cho toàn bộ lệnh, đó là sẵn sàng hoạt động, nó thậm chí còn có địa chỉ để kết nối. Nhân tiện, đây là lý do tại sao địa chỉ quảng cáo đó rất quan trọng khi bạn đang khởi tạo một cụm và cũng như khi bạn tham gia cùng những người quản lý khác vì người quản lý là các node được kết nối bởi nhân viên, vì vậy địa chỉ được quảng cáo cụ thể này có có sẵn cho họ, và rất may là trong trường hợp này. Được rồi, hãy sao chép nó, sau đó tôi cần chuyển ngữ cảnh ở đây và chúng tôi sẽ chuyển sang w2 và dán cái này vào. Bây giờ trước khi tôi chạy cái đó, tôi muốn nhắc bạn chú ý đến phía bên phải và phía trên và xem những gì xảy ra. Nhìn kìa. Thực sự nhanh chóng node w2 được tham gia vào cụm và đó là tất nhiên, bởi vì nó đã được khởi động, nhưng bạn có thấy tốc độ đó nhanh như thế nào không? Thật là ấn tượng.

Mô phỏng mất nguồn và quan sát trạng thái node khi các node trực tuyến trở lại

Bây giờ chúng ta hãy xem xét khi một cái gì đó đi theo hướng ngược lại. Hãy xem khi nào có sự cố. Bạn có thể nhận thấy điều này xảy ra nếu, ví dụ, bạn khởi động lại máy trong khi máy ảo của bạn đang chạy và chúng không còn chạy nữa, tôi sẽ mô phỏng điều đó với một lệnh tạm dừng mơ hồ, tương đương với lệnh dừng hoặc tắt máy. . Trong trường hợp này, tôi sẽ chỉ định w1 là node. Bây giờ tôi sẽ đánh dấu các phần của màn hình để xem ở đây khi máy ảo đó tắt. Nó đây. Bạn có thấy điều đó đi xuống không? Hãy làm công nhân 2 ở đây và xem điều gì sẽ xảy ra. Và bạn xem, bạn có thấy rằng, chúng tôi có w2 màu đỏ, và sau đó ở trên cùng, bạn có nhận thấy điều gì xảy ra không? Hơi tinh tế ngay cả khi sự khác biệt nhấp nháy, nhưng bạn có thể thấy trong cột TRẠNG THÁI cho w1 và w2, cả hai đều ở trạng thái Không hoạt động, trong khi trước đó chúng đã Sẵn sàng như m1. Và tất nhiên, chúng tôi có thể mô phỏng những thứ này trở lại trực tuyến bằng cách đưa chúng trở lại, w1, w2, hãy chờ nó ở đây. Hãy chú ý đến tác động đến hai quan điểm của chúng tôi. Này, w1 đã sẵn sàng, và w1 cũng đã sẵn sàng ở phía bên tay phải. Tuy nhiên, chấm đỏ vẫn ở đó cho w2, gần như ở đó. Được rồi, bạn có thấy sự khác biệt lóe lên ở phía trên và phía trên ở phía bên tay phải , mọi thứ đều xanh tươi trở lại. Vì vậy, đây là một trong những lý do tôi thực sự yêu thích Vagrant, khả năng mà bạn có thể chơi với các máy ảo, bật, tắt chúng, mô phỏng tất cả các loại điều kiện mà bạn sẽ rất khó để mô phỏng trong thực tế, thiếu có lẽ kéo phích cắm ra khỏi tường để mô phỏng sự cố mất điện. Đừng làm điều đó mặc dù. Điều tuyệt vời là bạn có các máy ảo và điều đó mang lại cho bạn một cách hay để xem điều gì xảy ra với cụm của bạn. Bạn có thể thử rất nhiều kịch bản khác nhau.

Tham gia node quản lý thứ hai

Tôi đang hiển thị một node nữa ở đây. Lần này tôi muốn tham gia một node người quản lý khác. Vì vậy, tôi đang đưa m2 lên. Tôi sẽ tạm dừng việc này và quay lại. Được rồi, vậy là xong. Và sau đó, trong tab mới ở đây, tôi đã chạy lệnh tham gia mã thông báo để lấy mã thông báo người quản lý vì nó khác với mã thông báo công nhân. Sao chép nó, và sau đó tôi sẽ quay trở lại cửa sổ đầu tiên của chúng tôi ở đây và xuống bên dưới. Hiện tại, tôi đang nhầm ngữ cảnh, vì vậy tôi sẽ cần chuyển ngữ cảnh sang m2. Không tìm thấy. Đó là bởi vì tôi chưa tạo ra nó. Hãy tạo điều đó thật nhanh chóng. Được rồi, trên hết, chúng tôi đã có một sự thay đổi lớn và có vẻ như mọi thứ đang chạy ngay bây giờ. Bây giờ chúng ta có thể chuyển sang bối cảnh m2 của chúng ta. Tuyệt vời! Và sau đó tôi sẽ có thể dán vào lệnh tham gia với mã thông báo cho mã thông báo người quản lý. Sau đó, chú ý bên trên và bên phải khi tôi làm điều này. Vì vậy, m2 mất nhiều thời gian hơn để trực tuyến và, đôi khi tôi nhận thấy điều này, Visualizer UI sẽ có thứ gì đó không ổn, chẳng hạn như dung lượng RAM ở đây. Chỉ cần làm mới nếu đúng như vậy và nó sẽ nhận bất kỳ thay đổi nào mà nó đã bỏ lỡ. Sau đó, ở phía bên tay trái , bạn có thể thấy mọi thứ đã sẵn sàng để lắc lư. Bây giờ về các node người quản lý, chúng ta có m1 là node dẫn đầu, và sau đó m2 là có thể truy cập được. Vì vậy, trong số các nhà quản lý, 1, nhà lãnh đạo, là người thực sự chịu trách nhiệm thực hiện công việc nặng nhọc. Những người khác ở đó để can thiệp nếu có vấn đề gì xảy ra và trở thành người dẫn đầu. Và để làm được điều đó, họ phải dành thời gian duy trì trạng thái của cụm một cách chính xác để có thể thực sự bước lên nếu thời gian đến. Ngoài ra, theo bản chất của việc có nhiều người quản lý và khả năng thất bại, bạn có khả năng thất bại, do đó giảm thời gian chết miễn là bạn có nhiều hơn một người quản lý và tối thiểu là ba người được khuyến nghị.

play-with-**docker**.com - Cách thiết lập Phòng thí nghiệm Học tập với Cụm nhiều node miễn phí trên mây!

Tóm lại, tôi muốn cung cấp cho bạn một khả năng khác nếu bạn muốn thiết lập nhiều node , bạn chỉ có sẵn **Docker** Desktop hoặc một số node duy nhất, và bạn không muốn phải xử lý các máy ảo. Chà, bạn đang gặp may. Có một công cụ tên là Play with **Docker** được phát hành cách đây nhiều năm, nó đã xuất hiện được một thời gian và nó thực sự rất hay. Tôi nhớ đã tổ chức toàn bộ hội thảo về điều này cho những người tham gia và đó thực sự là một cách tuyệt vời để mọi người thiết lập và điều hành với một nhóm. Vì vậy, chỉ cần ra ngoài để chơi - với - **docker**.com, với dấu gạch ngang giữa các từ, hãy nhấp vào node Bắt đầu và một phiên sẽ được bắt đầu. Các phiên này kéo dài 4 giờ. Vì vậy, một trong những hạn chế của việc này là nó tốt hơn cho các bài tập học tập ngắn hạn mà bạn có, không tốt cho lâu dài. Và tất nhiên, đó là để cố gắng hạn chế một số lạm dụng sẽ xảy ra nếu họ có một hệ thống trực tuyến mà bạn có thể lưu trữ các ứng dụng từ đó. Bởi vì bạn thực sự có thể truy cập các ứng dụng này trên web, mọi người sẽ bắt đầu chuyển hướng các yêu cầu đến chúng từ trang web của họ. Vì vậy, bạn có thể muốn dựa vào các tập lệnh hoặc sao chép / dán các đoạn mã mà bạn có thể sử dụng để tạo môi trường của mình. Hoặc, có một cờ lê nhỏ ở đây mà bạn có thể nhấp vào và nó bật lên các mẫu mà bạn có thể tạo. Bởi vì điều này có tác dụng gì, nó ít nhiều có thể tạo ra các phiên bản một cách nhanh chóng. Chúng là các phiên bản mô phỏng của máy móc. Vì vậy, tôi có thể nhấp vào node này lặp đi lặp lại và tôi nhận được ba node ở đây. Và bây giờ tôi có thể bắt đầu quá trình khởi tạo chúng bằng **docker** **Swarm** init và tham gia. Nhưng nếu tôi không muốn làm điều đó, tôi sẽ loại bỏ các node này ở đây, thay vào đó, tôi có thể đến đây và nói, này, tôi muốn mẫu 3 Người quản lý và 2 Người lao động này. Nói chung là đối với tôi. Làm mọi thứ cho tôi. Bạn có thể thấy tất cả các node được đặt tên độc đáo. Chúng tôi có manager1, 2 và 3. Chúng tôi có worker1 và worker2. Chúng ta có thể bắt đầu ngay lập tức, hãy xem ls **docker** node. Và bạn có thể thấy tất cả các node của chúng tôi được kết nối với nhau, vì vậy việc khởi tạo nhóm và tham gia các node đều được thực hiện cho bạn. Vì vậy, một cách tốt là chỉ cần đứng dậy và chạy với một bài tập rất cơ bản và thậm chí còn nhanh hơn để tạo ra một bầy cho chính mình. Mặc dù vậy, bạn có thể tự động hóa quá trình này với một cái gì đó như Vagrant. Bạn có thể có một tập lệnh trong đó các node được nối với nhau. Điều đó không khó để viết, và nó thậm chí có thể là một bài tập nhỏ thú vị nếu bạn muốn thử thách.

Thúc đẩy, hạ cấp và kiểm tra các node

Tại sao chúng ta không lấy nó để lái thử và thử một vài thứ mà chúng ta chưa làm được. Hãy xem xét node, quảng cáo và hạ cấp. Về cơ bản, người lao động chạy các ứng dụng của chúng tôi, người quản lý điều phối những người đang chạy ứng dụng của chúng tôi và theo mặc định, người quản lý cũng có thể sử dụng tài nguyên của họ để chạy các ứng dụng. Mặc dù vậy, khi bạn xây dựng một cụm lớn hơn, bạn sẽ muốn loại bỏ khả năng đó khỏi những người quản lý đó. Điều đó nói lên rằng, có thể một lúc nào đó bạn cần thăng chức một nhân viên lên một người quản lý, chẳng hạn, một trong những người quản lý của bạn bị mất vì một lý do nào đó. Để duy trì số đại biểu, bạn có thể cần đưa một người quản lý mới vào cuộc. Hãy chỉ xem quá trình đó trông như thế nào. Vì vậy, tôi có một node **Docker** ở đây. Nếu chúng ta xem xét lệnh node **docker**, có sẵn các lệnh con khác. Bây giờ chúng ta đã xem xét lệnh phụ ls. Chúng tôi cũng có một lệnh phụ quảng cáo. Vậy tại sao chúng ta không lấy worker1 và thăng cấp worker1 thành một node quản lý. Vì vậy, chỉ cần mũi tên lên ở đây, và tôi sẽ gõ quảng cáo và hoàn thành tab là một điều tuyệt vời khi nói đến **Docker** CLI. Nếu bạn chưa cài đặt cái đó, hãy đảm bảo rằng bạn cài đặt cái đó cho môi trường cụ thể của mình, nó làm cho cuộc sống dễ dàng hơn rất nhiều với tất cả các lệnh sub sub sub sub sub này. Vì vậy, tôi có thể chạy nó, nhận một số trợ giúp cho nó và bạn có thể thấy chúng tôi chỉ cần chỉ định node nào chúng tôi muốn quảng cáo, vì vậy đó là hướng một chiều để quảng cáo. Chúng ta không cần chỉ định bất kỳ đối số nào ở đây. Hãy chỉ cần lấy worker2 và thăng chức lên người quản lý. Và đó là cách dễ dàng để thăng chức lên một người quản lý. Và bạn có thể thấy ở phía bên tay trái , có vẻ như người quản lý được đánh dấu bằng một biểu tượng hình người nhỏ, gần giống như có lẽ một người giám sát đang theo dõi bạn. Và nếu tôi thực hiện một node **docker** ls, tôi có thể xác nhận rằng điều đó đã thực sự xảy ra, chương trình khuyến mãi, có thể thấy worker2, không còn là tên chính xác nhất nữa, nhưng đủ rồi, bây giờ có thể truy cập được với tư cách là trình quản lý dự phòng. Nếu tôi muốn biết thêm một số chi tiết, nhân tiện, Ctrl + L sẽ xóa điều này ở đây và lấy node **Docker** và sau đó kiểm tra như một lệnh phụ, và sau đó, tôi có thể cung cấp cho worker2, rất nhiều thông tin khác về worker này ở đây, và tôi sẽ có thể cuộn lên và thấy ở đâu đó trong đây, của bạn đây, tôi có vai trò được chỉ định là người quản lý và tính khả dụng của nó đang hoạt động. Và tất nhiên, nếu tôi muốn làm ngược lại, có lẽ tôi đã sửa được trình quản lý đang ngoại tuyến và tôi muốn chuyển đổi worker2 trở lại chỉ là worker và nó chỉ là node **docker** và sau đó chúng ta có lệnh giáng cấp ở đây . Và một lần nữa, không có tham số nào khác ngoài tên mà chúng tôi muốn hạ cấp, và bạn xem, nó đã bị hạ cấp. Hãy xem ls node của chúng ta và bạn có thể thấy nó không còn trạng thái người quản lý nữa, và do đó, nó chỉ là một node công nhân. Và nếu bạn muốn xác nhận rằng bằng cách kiểm tra, chúng ta có thể chỉ cần mũi tên lên ở đây, cuộn lên trên cùng ở đây và một lần nữa, chúng ta có thể xem xét thông số kỹ thuật và chúng ta có thể thấy rằng vai trò ở đây là công nhân.

Kết nối với play-with-**docker**.com qua SSH - Và thậm chí còn tốt hơn, Ngữ cảnh **Docker** dựa trên SSH!

Một điều nữa trước khi tôi tiếp tục, có một node nhỏ ở đây nơi bạn có thể nhấp để truy cập vào môi trường này, nếu bạn muốn truy cập vào nó, vì vậy bạn có thể sao chép nó, bạn có thể truy cập ngay vào dòng lệnh của tôi và dán đó ở đây. Tôi được nhắc chấp nhận dấu vân tay và bây giờ tôi được kết nối thậm chí không cần mật khẩu, nhưng nếu bạn nghĩ về điều đó, khoảng thời gian dài lớn mà bạn phải thêm vào để kết nối với node này là đủ về mặt mật khẩu cho mục đích của việc học. Vì vậy, về mặt nào đó, bạn thực sự có thể có quyền truy cập vào hệ thống tệp cục bộ của mình và làm việc với một node từ xa qua ssh trong trường hợp này. Trên thực tế, nếu bạn thực sự muốn thử thách, tôi chưa thử điều này, nhưng chúng ta có thể thử tạo ngữ cảnh **docker** cho việc này, đặt máy chủ lưu trữ thành lệnh ssh đã sao chép đó, quay lại đây và chúng tôi sẽ thay đổi điều này để cho biết giao thức là ssh, vì vậy: //, hãy đặt tên pwd cho Play with **Docker**. Vì vậy, chúng ta hãy cố gắng kết nối với điều đó. Vì vậy, chúng tôi sử dụng pwd và node ls. Này, nhìn này, nó cũng hoạt động. Vì vậy, bây giờ tôi đang ở trên máy tính thực tế của mình, tôi có sẵn tất cả các lệnh mà tôi quen thuộc với tôi. Tôi thậm chí không phải ssh vào node đó. Nhưng đây là một tính năng thực sự thú vị, nó cung cấp cho bạn khả năng làm việc với các tệp trên hệ thống tệp cục bộ của bạn, vì vậy bạn sẽ không phải sao chép và dán mọi thứ vào môi trường khác đó. Nhưng hãy nhớ rằng phiên này sẽ hết hạn. Trên thực tế, trong thời gian tôi cho bạn thấy điều này, tôi đã mất 14 phút trong 4 giờ của mình. Vì vậy, chỉ cần lưu nội dung trong một tệp và sẵn sàng sử dụng nếu bạn muốn thử một cái gì đó mới một lần nữa. Tuy nhiên, điều thú vị là sau khi hoàn thành, bạn có thể nhấn ĐÓNG PHẦN và mọi thứ đều được xóa. HIỂU RỒI. Nó đưa bạn trở lại trang bắt đầu, bạn có thể nhấn Bắt đầu lại bất kỳ lúc nào bạn muốn và sau đó bạn có một phiên hoàn toàn mới ở đây, bạn có thể thêm phiên bản mới của mình, theo cách này, nếu bạn muốn dùng thử có thể nối các node với nhau theo cách thủ công. Trong khi tôi đang ở đó, có lẽ tôi nên thoát khỏi bối cảnh đó, coi như phiên đó đã kết thúc.

Triển khai dịch vụ cho cụm

Tóm tắt những thách thức của việc triển khai vùng chứa nhiều node với các dịch vụ

Bây giờ chúng ta đã dành một chút thời gian để nói về các node và tôi đã nói rằng có một sự phân chia tốt đẹp giữa các node và sau đó là khía cạnh lớn khác của những gì chúng ta chạy trên một cụm. Chúng tôi có một số thiết lập cụm khác nhau được thiết kế ở đây với nhiều công nghệ khác nhau. Bạn có thể chọn những gì bạn thích bởi vì bây giờ chúng tôi sẽ tập trung vào những gì chúng tôi chạy trên đầu các node đó. Và điều đầu tiên chúng tôi muốn xem là cách chúng tôi có thể sắp xếp các volume. Cũng giống như trên một node độc lập, chúng tôi muốn tạo ra các ứng dụng đang chạy trong các bong bóng nhỏ cô lập. Vì vậy, ví dụ, nếu hai ứng dụng có cùng phụ thuộc thư viện, chúng không xung đột với nhau. May mắn thay, trong trường hợp này, với một bầy, không có xung đột nào về các vùng chứa chức năng của một vùng chứa bởi vì mỗi vùng chứa đó có một image mà nó được xây dựng trên đó bị cô lập bất kể nó nằm trên cùng một node hay một node khác. Nhưng sau đó có những thách thức khác khi nói đến những gì chúng tôi muốn chạy trên cụm của mình. Ví dụ: tôi đã đề cập có lẽ hai vùng chứa trên cùng một máy chủ cần truy cập vào một cổng và có lẽ cả hai đều là dịch vụ web và có thể chúng đều muốn sử dụng cổng web. Chà, chúng tôi biết rằng các vùng chứa giải quyết được phần nào vấn đề đó bằng cách cung cấp cho mỗi ứng dụng này giao diện ảo của riêng chúng với mạng, vì vậy hãy kết hợp với mỗi cổng 80 nếu chúng muốn. Nhưng sau đó nếu hai ứng dụng này cần phải nói chuyện với nhau, bằng cách nào đó chúng phải được định tuyến cùng nhau và chúng ta có thể làm điều đó nếu chúng nằm trên một node duy nhất. Nhưng điều gì sẽ xảy ra nếu họ cần nói chuyện với nhau và họ đang ở các node khác nhau? Trong trường hợp đó, chúng ta phải tạo ra một số loại mạng ảo trải dài qua các node của chúng ta để cung cấp kết nối cho các ứng dụng của chúng ta. Tất nhiên, một vấn đề cơ bản hơn nếu tôi muốn chạy một vùng chứa khác, nó sẽ được đặt ở đâu? Nó có nên đến đây không? Nó có nên đến đây không? Và nó cần phải phát triển về nguồn lực mà nó sử dụng, có không gian cho điều đó ở đây không? Có lẽ nó cần phải đến đây để thay thế. Hoặc điều gì ở đây, nơi nó đá ra một số container khác? Và làm thế nào để chúng tôi làm việc với công cụ để nó có cùng một quy trình làm việc độc lập, nơi chúng tôi tiếp tục thêm các vùng chứa và chúng tôi không thực sự quan tâm đến mối quan tâm về vị trí này? Tất nhiên, chúng tôi có thể hết tài nguyên trên một node duy nhất. Đó là lý do tại sao chúng ta đang nói về nhiều node. Nhưng chúng tôi muốn giải pháp tương tự như với một phiên bản **Docker** độc lập. Một yếu tố khác, chính xác thì chúng ta đang chạy ở đây là gì? Có lẽ là một ứng dụng web duy nhất. Và chính xác thì nó được phân phối như thế nào? Có lẽ hai trường hợp riêng biệt. Nó có cần một số loại cơ sở dữ liệu không? Và làm thế nào mà nó được phân phối để chúng ta có chuyển đổi dự phòng? Có lẽ là một chế độ chờ trong một trường chính. Có lẽ rủi ro hợp lý đòi hỏi một số loại mô hình được lưu trong bộ nhớ và nếu nó đủ phức tạp, nó có thể yêu cầu một số loại API ngồi trước nó được phân phối trên toàn bộ cụm cũng như để xử lý tải mà ứng dụng web cho vay nói chuyện với sau đó nói chuyện với mô hình rủi ro. Bởi vì những gì sắp xảy ra là chúng tôi có ai đó đi cùng nói rằng, này, tôi cần một chiếc ô tô. Vì vậy, họ đến để vay, và họ đưa ra yêu cầu về một ứng dụng. Ứng dụng web có thể cần cơ sở dữ liệu của nó để đọc một số thông tin, vì vậy nó có thể đọc từ chế độ chờ ngay bên cạnh. Và sau đó, có thể bạn cần một số thông tin về rủi ro liên quan, và vì vậy cần tìm API rủi ro đó ở đâu. Ồ, có một ví dụ tôi có thể sử dụng. Được chứ. Và sau đó, API rủi ro đó có thể cần phải nói, này, các mô hình của tôi ở đâu? Tất cả những câu hỏi này được hỏi là một dạng khám phá dịch vụ. Ngay cả khi đó là một người bên ngoài tổ chức, đó là một truy vấn bên ngoài về nơi họ có thể đến để điền vào đơn xin vay. Hoặc nó có thể là một truy vấn nội bộ về cách tìm một dịch vụ khác đang chạy bên trong một cụm. Đây chỉ là một số thách thức mà **Docker** phải giải quyết và minh bạch cho chúng tôi, và các giải pháp cho phía chứa của phương trình được gói gọn trong cái được gọi là dịch vụ. Vì vậy, chúng ta hãy xem triển khai một dịch vụ như thế nào và so sánh nó với triển khai một vùng chứa.

Full Auto **Swarm** với Trình hiển thị vùng chứa độc lập để chuyển đổi thành dịch vụ

Được rồi, tôi đang ngồi bên trong một thư mục autos bên trong thư mục labs. Xem đường dẫn ngay tại đó và bạn có thể xem các tệp bên trong đây. Có một tệp Vagrantfile ở đây. Và tôi muốn cho bạn thấy điều này bởi vì trong quá trình tìm hiểu một dịch vụ, tôi đã thiết lập một môi trường Vagrant khác hoàn toàn tự động, bao gồm việc thêm phiên bản chứa trong bộ hiển thị đó và sau đó tôi muốn đối chiếu điều đó với phiên bản tương đương dưới dạng dịch vụ. Vì vậy, điều đầu tiên tôi sẽ làm, tôi muốn xem xét Vagrantfile một cách ngắn gọn. Vì vậy, trong autos và repo, có một Vagrantfile đáng chú ý nhất. Thực sự thay đổi lớn ở đây là tôi đã thêm các lệnh gọi vào cung cấp, và sau đó tôi thêm các tập lệnh cho những người đó bên dưới thư mục tập lệnh. Một tập lệnh sẽ khởi tạo bầy đàn với m1 khi nó trực tuyến và nó sẽ ghi ra hai mã thông báo, mã thông báo công nhân và mã thông báo người quản lý ra đĩa. Và sau đó khi các node khác trực tuyến, các tập lệnh worker - join và manager - join được chạy để tham gia các node đó vào cụm tự động và chúng sử dụng các mã đã được viết ra. Nuke - tokens chỉ là lời nhắc nhở sau thực tế. Bạn có thể muốn xóa chúng. Không cần thiết vì chúng sẽ bị ghi đè nếu bạn khởi động lại bầy đàn. Và điều quan trọng nhất trong số này là tạo trình hiển thị. Tôi sẽ có thể truy cập vào dòng lệnh ở đây, và tôi sẽ hơi mơ hồ một chút, và m1 cần được tạo trước trước bất kỳ người nào khác. Vì vậy, các mã thông báo được ghi vào đĩa để phần còn lại của chúng có thể tham gia thông qua cụm. Nhân tiện, trong khi điều này đang khởi động, trong trình duyệt, tôi đã mở một tab và chỉ nó vào 101, đó là IP. Tôi đã thay đổi địa chỉ IP từ 201 và 211 và 212, v.v. Tôi đã thay đổi chúng thành phạm vi 100. Tôi muốn tải trước tab này để có sẵn trình hiển thị thứ hai, chúng ta có thể xem nó. Khi **Docker** hoàn thành, nó sẽ chuyển sang phần cung cấp hoặc tạo ra điều đó và tôi muốn có thể thấy điều đó trước khi nhóm được khởi chạy. Và điều đó sẽ xảy ra thực sự nhanh chóng, vậy tại sao chúng ta không chia đôi màn hình ở đây. Này, hãy nhìn vào đó, và m1 đã được khởi tạo. Đó là cách điều đó xảy ra nhanh chóng. Đối với phần còn lại của các node. Và nhân tiện, tôi đã thực sự bắt đầu với người lang thang một mình, vì vậy tất cả các node đang bắt đầu ngay bây giờ. M2 thực sự đang trong quá trình. Để đảm bảo rằng m1 đã hoàn thành, tôi đang chạy tuần tự để m1 hoàn thành, sau đó là m2, sau đó là m3, v.v. Và hãy để tôi tăng tốc độ ở đây. Bây giờ hãy nói về việc biến vùng chứa độc lập này thành một dịch vụ.

Tính đối xứng của **Docker** Container Run - trợ giúp và **Docker** Service Create --help

Vì vậy, chúng tôi muốn chuyển đổi câu lệnh chạy bộ chứa **docker** của chúng tôi thành câu lệnh tương đương với một dịch vụ, một dịch vụ là đại diện của một đơn vị tính toán. Một phần lý do tại sao các vùng chứa được gọi là dịch vụ bây giờ là vì có thể có nhiều hơn một vùng chứa cho mỗi dịch vụ. Vì vậy, một vùng chứa là một khái niệm cấp thấp hơn vẫn còn tồn tại khi nói đến giao dịch với **Docker** **Swarm**. Một khái niệm cấp cao hơn là một dịch vụ. Một điều tôi thích làm là chỉ cần xem qua dòng lệnh. Vì vậy, trước tiên, chúng tôi có bộ chứa **docker**, và bạn có thể đã thấy cái này. Đó là một danh sách dài rất dài bao gồm toàn bộ các lệnh. Đáng chú ý, run là cách chúng ta có thể tạo và khởi động một vùng chứa trong một thao tác. Nếu chúng ta nhìn vào đó, bạn có thể thấy có rất nhiều lựa chọn để vượt qua. Tôi khuyên bạn nên đọc qua điều này tại một số điểm nếu bạn chưa sử dụng lệnh chạy bộ chứa **docker**, nhưng điều quan trọng hơn là chuyển sang phần dịch vụ của **docker** của mọi thứ. Điều này được viết tắt nhiều hơn về mặt hoạt động. Mặc dù nó đã phát triển một chút theo thời gian, nhưng tôi nghi ngờ là nó sẽ dài ra bất thường như các danh sách khác vì hầu hết những gì cần thiết về đại diện cho một dịch vụ đã được tập hợp các lệnh con này nắm bắt. Có thể cho rằng, điều này có thể được thiết kế tốt hơn so với các lệnh con vùng chứa **docker** ban đầu bởi vì chúng xuất phát từ những ngày đầu của **Docker**. Ngoài ra, điều này đại diện cho một khái niệm hơi khác ở chỗ chúng ta đang thực sự nói về định nghĩa cho một hoặc nhiều vùng chứa vào cuối ngày. Bây giờ nếu bạn phải đoán xem bạn nghĩ tùy chọn nào trong số này gần tương đương với những gì chúng ta đã thấy khi chạy bộ chứa **docker**, thì đó là lệnh tạo ở trên cùng. Bây giờ nó hơi khác một chút. Chúng ta phải tạo lệnh bằng **docker** container và nó không làm điều tương tự, nhưng tạo ở đây sẽ tạo ra dịch vụ của chúng ta, định nghĩa nó và sau đó **Swarm** được phép sử dụng định nghĩa dịch vụ đó và biến nó thành hiện thực. Vì vậy, chúng ta hãy xem xét sự trợ giúp cho việc này. Và bạn có thể thấy có một danh sách dài các tùy chọn ở đây. Và trên thực tế, nếu bạn so sánh hai bên cạnh nhau, bạn sẽ tìm thấy rất nhiều điểm tương đồng giữa điều này và những gì chúng ta đã có với việc chạy vùng chứa **docker**. Đây là một sự khác biệt ở đây của cả hai kết quả đầu ra. Tôi có một tập lệnh nhỏ trong kho lưu trữ nếu bạn muốn tự mình thử điều này. Và như bạn có thể thấy, màu vàng, là phần chồng lên nhau, có khá nhiều và sau đó là màu đỏ. Thông thường, bạn sẽ thấy rằng đôi khi một cái gì đó đã được đổi tên hoặc văn bản mô tả đã được thay đổi một chút, và đó là một số khác biệt ở đây. Nhưng như bạn có thể thấy, có khá nhiều điểm trùng lặp. Một lần nữa, đây là danh sách mà tôi khuyên bạn nên đọc qua hoặc một lúc nào đó hãy ngồi xuống và xem qua. Ví dụ: có nhiều cách để kiểm soát thiết lập DNS bên trong vùng chứa của bạn. Bạn không cần biết cài đặt chính xác mà chỉ cần biết rằng bạn có thể chỉnh sửa DNS. Sau đó, khi bạn cần làm như vậy, bạn có thể đến tìm các tùy chọn liên quan.

Chuyển và chạy một dịch vụ Visualizer

Được rồi, vì vậy, để tiết kiệm thời gian, tôi đã có một cổng đóng hộp sẵn của container chạy cho dịch vụ viz của chúng tôi thành một dịch vụ thực tế, và đó là bên trong dịch vụ chuyển đổi - viz - sang - này . Nó nằm ngay bên cạnh tập lệnh shell shell m1 - viz - standalone - container. Vì vậy, trước tiên, chúng tôi tạo dịch vụ **docker** thay vì chạy bộ chứa **docker** của chúng tôi. Chúng tôi có một đội ở đây. Chúng tôi có thể thêm điều đó xuống đây nếu chúng tôi muốn. Tôi đã chọn để bỏ nó đi. Để xuất bản cổng, về cơ bản tôi vừa sao chép cùng một giá trị vào tập lệnh bên dưới. Để đảm bảo rằng điều này không gây ra bất kỳ xung đột nào, tại sao chúng ta không chuyển cái này thành 81, vì vậy cái này sẽ được ánh xạ thành 8081. Vì chúng ta vẫn còn một cái độc lập đang chạy, chúng ta có thể để nó chạy. Và sau đó thay vì cờ ‑‑ volume, có một cờ ‑‑ mount. Và điều này dài dòng hơn một chút bằng cách sử dụng định dạng dài hoặc cú pháp, để chỉ định các tùy chọn về loại ổ đĩa chính xác đang gắn kết. Vì vậy, ở trên đây, đây hoàn toàn là một liên kết ràng buộc chỉ định một đường dẫn máy chủ và sau đó là một đường dẫn đích bên trong vùng chứa. Chà, ở bên dưới, chúng ta có cùng thông tin. Chúng ta nói type = bind, sau đó chúng ta có nguồn trên máy chủ lưu trữ và sau đó đích của chúng ta là bên trong vùng chứa. Phần cuối cùng của điều này sau đó chỉ là image của chúng tôi. Và bây giờ trong khi bạn đang chuyển đổi nó, làm thế nào tôi biết, ví dụ, làm thế nào tôi có thể biết rằng tôi phải sử dụng hoặc tôi có thể tiếp tục và sử dụng cùng một giá trị để xuất bản? Chà, ngoài trợ giúp về dòng lệnh, còn có một bộ tài liệu tuyệt vời xoay quanh việc tạo dịch vụ **docker** và sau đó là lệnh chạy **docker**. Bây giờ việc chạy vùng chứa của **docker** cũng giống như vậy, nhưng điều này được nhúng bên trong phần chạy của **docker** và Tham chiếu. Hãy chắc chắn rằng bạn tìm Tham khảo trên tài liệu **Docker**. Khi bạn tìm thấy các lệnh quan trọng, chúng sẽ có cả đống tài liệu kèm theo. Ví dụ, tạo một vùng chứa và chạy nó. Nhìn vào tất cả các thông tin. Chúng ta có thể xem qua rất nhiều lời giải thích bên trong đây. Ví dụ: đây là cách chúng ta có thể xử lý khối lượng, liên kết gắn kết, sử dụng cú pháp mount thay vì thực sự bằng lệnh chạy vùng chứa. Và mặt trái của mọi thứ, trong quá trình tạo dịch vụ, chúng ta có thể tìm kiếm một phần tương đương, và chúng ta bắt đầu. Chúng tôi đã có Thêm liên kết gắn kết, ổ đĩa hoặc hệ thống tệp bộ nhớ. Và dưới đây là nơi bạn có thể xem cách các khối lượng được chuyển thành thú cưỡi. Và nếu bạn cuộn xuống đây, bạn sẽ thấy một số tùy chọn mà tôi vừa nói đến. Ví dụ, loại gắn kết, liên kết gắn kết trong trường hợp của chúng ta, nguồn, đích, những điều này có nghĩa là gì. Thậm chí có một lập luận chỉ đọc mà chúng tôi chưa sử dụng. Và sau đó sẽ có một cái gì đó ở đây cho các cổng. Đúng vậy, nếu chúng ta cuộn xuống đây, sẽ có một phần dành cho các cổng và điều này sẽ giải thích rằng chúng ta đang gắn bó với cú pháp dạng ngắn. Cũng có một cú pháp biểu mẫu dài cho điều này, như cú pháp biểu mẫu dài gắn kết, nhưng trong trường hợp này chúng ta có thể sử dụng một trong hai, và vì vậy tôi sẽ gắn bó với những gì chúng ta đã có ban đầu. Vì vậy, điều tiếp theo là với sự trợ giúp đó là chúng ta có dòng lệnh và trực tuyến. Chúng ta có thể đi qua đối số và quyết định cách chúng ta muốn chuyển mọi thứ qua hoặc cách chúng ta phải chuyển mọi thứ và tìm ra một lệnh mới để sử dụng. Và sau đó, bạn có thể sao chép nội dung ở đây của tệp này hoặc bạn có thể thực thi tập lệnh. Tôi sẽ chia màn hình ở đây. Trên hết, tôi đã có Trình hiển thị độc lập của mình để chúng ta có thể xem những gì đang diễn ra ở đây. Và sau đó ở phía dưới, tôi sẽ tiếp tục và dán vào lệnh mà tôi đã sao chép. Và hãy tắt nó đi và xem điều gì sẽ xảy ra. Vì vậy, bởi vì chúng tôi không tách ra, chúng tôi sẽ nhận được một số thông tin về tiến trình triển khai dịch vụ này cho cụm của chúng tôi. Và nếu bạn nhìn vào đó, dịch vụ của chúng tôi vừa xuất hiện trên m1. Đầu tiên, chúng ta hãy kiểm tra và xem liệu nó có hoạt động hay không. Tôi sẽ lên đây trên cùng, tạo một tab mới. Bạn có biết địa chỉ IP nào tôi cần đi qua đây và cổng nào tôi cần sử dụng không? Vì vậy, đây là m2, vì vậy nó sẽ là 102, và sau đó nó là 8081 là cổng mà chúng tôi đang sử dụng. Và này, hãy nhìn vào đó. Bây giờ chúng tôi có trình hiển thị của chúng tôi đang chạy như một dịch vụ thay vì như một vùng chứa độc lập. Trên thực tế, chúng tôi có cả hai trong số họ đang hoạt động ngay bây giờ. Xem, có thể làm mới cả hai. Chúng tôi không cần cả hai, nhưng có vẻ như mọi thứ hầu như đã hoạt động. Bây giờ bạn có thể đã không gặp may mắn như tôi đã làm vì có một phần thông tin bị thiếu khi tôi chuyển kịch bản đó qua, và tôi đã làm điều đó một cách miễn cưỡng. Đó là một lập luận ngầm không dễ hiểu. Tham gia cùng tôi video tiếp theo để tôi có thể nói về điều đó.

Cập nhật dịch vụ **Docker** - buộc thay đổi vị trí mà không cần thay đổi dịch vụ của chúng tôi

Như tôi đã đề cập, một cái gì đó bị thiếu có thể ngẫu nhiên gây ra vấn đề cho chúng tôi. Điều đầu tiên tôi muốn làm để giải thích điều này là gì sẽ kích hoạt tình huống gây ra sự cố bởi vì ngay bây giờ với dịch vụ trình hiển thị này đang chạy trên một trong các node trình quản lý của chúng tôi, nó sẽ không phải là vấn đề. Thay vào đó, tôi cần làm cho điều này hoạt động trên một trong các node công nhân của chúng tôi. Bây giờ, có nhiều cách tôi có thể làm để làm điều đó. Một điều tôi sẽ làm là chỉ cho bạn cách bạn có thể kích hoạt các tác vụ và vùng chứa của mình được đặt ở nơi khác, nếu bạn muốn, mà không cần thực sự thay đổi bất kỳ điều gì về các dịch vụ đã triển khai. Qua dòng lệnh, điều đầu tiên tôi muốn làm là chạy bản cập nhật dịch vụ **docker**. Tôi sẽ không thay đổi bất cứ điều gì mặc dù. Thay vào đó, tôi cần phải cung cấp cho điều này, tôi đoán chúng ta có thể nhận được sự giúp đỡ và xem xét ở đây. Luôn luôn là một cách tốt để tìm ra những gì bạn có thể làm với một lệnh nhất định. Tôi cần cung cấp dịch vụ và tôi cũng muốn cung cấp một trong vô số đối số này. Đây là lập luận mà tôi cũng muốn áp dụng. Vì vậy, ngay cả khi không có thay đổi đối với dịch vụ, tôi muốn buộc nó phải được cập nhật như thể có. Hãy tưởng tượng nếu tôi triển khai một phiên bản mới của trình hiển thị, chẳng hạn. Vì vậy, tôi có thể chạy bản cập nhật dịch vụ **docker** của mình và sau đó tôi cần cung cấp cờ ‑‑ force, và sau đó tôi sẽ chỉ định viz. Khi điều này xảy ra, bạn nghĩ điều gì sẽ xảy ra? Chà, có thể đã cho nó đi, nhưng khi tôi làm điều này, điều này sẽ kích hoạt, hy vọng, một sự thay thế của nhiệm vụ và vùng chứa đó trên một node khác. Và tôi đang nói nhiệm vụ và vùng chứa, và tôi biết tôi vẫn chưa giải thích những điều đó. Nếu bạn muốn, một nhiệm vụ là trạng thái mong muốn cho vùng chứa đang hoạt động ở đâu đó trên cụm của bạn. Vì vậy, những gì tôi sẽ làm để xem liệu điều này có hiệu quả hay không là chuyển từ trình hiển thị dựa trên dịch vụ sang trình hiển thị mà chúng tôi vừa chạy độc lập trên node người quản lý của mình và xem nó hiển thị gì về vị trí của dịch vụ vì đây là một không phải là một dịch vụ, vì vậy nó có thể hoạt động độc lập với cách cụm thực sự được kết hợp với nhau. Hãy nhớ rằng cái này trên cổng 8080 Được rồi, vì vậy bạn có thể thấy rằng nó thực sự được triển khai lại cho cùng một node, có nghĩa là node này sẽ hoạt động trở lại. Vâng, được rồi. Vì vậy, chúng ta hãy thử điều đó một lần nữa. Đôi khi bạn phải làm điều đó một vài lần để điều này thực sự di chuyển xung quanh, nếu bạn muốn. Vì vậy, tôi sẽ quay lại với cái độc lập. Chỉ cần xem những gì xảy ra ở đây. Trời ạ, có vẻ như tôi không gặp may ở đây. Hãy quay bánh xe một lần nữa. Được rồi, lần thứ ba là cái duyên. Bạn có thể thấy chúng tôi đã chuyển sang công nhân 3.

Giám sát tác vụ CLI với Dịch vụ Watch **Docker** ps - định dạng

Bây giờ tôi phải nói điều này khá là mỉa mai. Tôi đã tắt máy tính của mình vào đêm qua và khi tôi bật lại, khi các quản lý trực tuyến, sau đó họ tiếp tục và sắp xếp lại công việc cho chính mình. Vì vậy, một lần nữa, viz đang ngồi đây chạy trên một người quản lý, đó không phải là điều tôi muốn. Nhưng điều đó tốt vì tôi thực sự muốn cho bạn thấy một giải pháp thay thế cho cập nhật bắt buộc trên một dịch vụ về việc đặt nhiệm vụ của bạn ở một nơi khác. Và đồng thời, chúng tôi sẽ bỏ giao diện người dùng. Tôi muốn chỉ cho bạn một cách khác tại dòng lệnh mà không có giao diện người dùng này. Và để làm điều đó, tôi muốn cho bạn thấy lý do tại sao tôi sẽ làm những gì tôi sẽ làm ở đây. Tôi muốn SSH vào Vagrant VM và thực hiện một lệnh trực tiếp. Và lý do cho điều đó là, tôi sẽ cho bạn thấy, chúng tôi sẽ chạy thời gian về điều này ở đây, dịch vụ **docker**, và chúng tôi sẽ thực hiện một ps. Ps đang lắng nghe các nhiệm vụ của bạn, hoặc bạn có thể coi chúng như các volume. Hãy nhớ rằng, chúng là trạng thái mong muốn cho các vùng chứa của bạn, vì vậy chúng có thể chưa tồn tại và nhiệm vụ là khái niệm đại diện cho những gì vùng chứa của bạn sẽ trở thành sau đó. Dù sao, nếu tôi chạy cục bộ này, thì phải mất khoảng nửa giây để chạy. Nếu tôi chạy đi chạy lại điều này, mỗi lần sẽ mất nửa giây. Và tôi muốn chạy điều này ít nhất là khoảng phụ thứ hai nếu có thể. Vì vậy, những gì tôi muốn cho bạn thấy là nó sẽ như thế nào khi chạy lệnh này ngay trên chính trình quản lý. Vì vậy, tôi sẽ SSH vào m1, và tôi sẽ chạy cùng một dịch vụ **docker** ps và sau đó viz như một dịch vụ. Nhưng điều quan trọng là nếu tôi chạy thời gian trên đầu, bạn có thể thấy điều này nói chung là nhanh hơn rất nhiều. Chúng tôi đang xem xét tốc độ nhanh gấp 10 lần. Vì vậy, tôi thực sự không thể chạy ở các khoảng phụ thứ hai ở đây vì những gì tôi muốn làm là chạy lệnh watch. Và với lệnh này, tôi sẽ chuyển một số đối số. Tôi sẽ ẩn tiêu đề, - d để làm nổi bật sự khác biệt, và sau đó chúng tôi sẽ thực hiện 0,5 trên khoảng thời gian. Vì vậy, điều đó sẽ không đánh thuế CPU của tôi là xấu. Và sau đó tôi sẽ chạy lệnh chính xác như trước đây, và sau đó chúng ta có thể xem ở đây khi các thay đổi xảy ra. Nhưng một điều cuối cùng tôi muốn thực hiện sửa đổi đối với lệnh này và giúp bạn hiểu một chút về những gì có sẵn ở đây cho bạn, tôi có thể thu nhỏ ở đây và điều đó có thể sẽ hiệu quả, nhưng điều đó thực sự nhỏ để bạn thấy, và tôi muốn bạn có thể thấy những gì đang xảy ra ở đây. Vì vậy, những gì tôi sẽ làm là cắt bớt các cột mà tôi không muốn. Chúng ta hãy xem xét điều đó tiếp theo.

JSON và đầu ra bảng tùy chỉnh với **Docker** Service ps --format

Vì vậy, tôi sẽ chỉ cần Ctrl + C để thoát khỏi đồng hồ. Vì vậy, hãy thử lại điều này ở đây, dịch vụ **docker** ps và viz, và tôi sẽ bỏ đồng hồ ngay bây giờ chỉ để tránh bất kỳ sự mơ hồ nào. Tôi sẽ thêm cờ định dạng ‑‑ ở đây và tôi sẽ dán vào một chuỗi. Chúng ta bắt đầu. Và tôi sẽ chạy nó, và sau đó tôi miễn cưỡng bỏ qua một thứ để tôi có thể nói về điều này. Bạn có thể thấy chúng tôi có một bảng đẹp ở đây, nhưng chúng tôi không có bất kỳ tiêu đề nào trên các cột. Nếu bạn muốn đưa những thứ đó vào đó và bạn muốn bảng được định dạng đồng đều, bất kể khoảng cách giữa các giá trị của các cột, hãy ném bảng từ lên phía trước tại đây, và bạn sẽ nhận lại các tiêu đề cột của mình và các cột nhất quán đẹp. Vì vậy, những gì tôi đã làm là tôi đã cắt bỏ hai cột một cách hiệu quả. Tôi đã cắt bỏ image, image này chiếm nhiều dung lượng và không thực sự quan trọng ngay bây giờ vì chúng tôi không thay đổi nó. Và tôi đã cắt các cổng. Mọi thứ khác là như nhau. Và, nhân tiện, nếu bạn tò mò về những gì chính xác bạn có thể bao gồm trong chuỗi định dạng đó, bạn chắc chắn có thể tham khảo tài liệu. Bạn cũng có thể chuyển một chuỗi định dạng với JSON và một dấu chấm bên trong mẫu. Và sau đó tôi cũng chuyển sang jq ngay ở cuối ở đây để làm cho nó được định dạng độc đáo. Tôi đã cài đặt JQ trên máy ảo như một phần của quá trình cấp phép của mình. Vì vậy, khi tôi chạy nó, tôi sẽ nhận được một danh sách tuyến tính, nếu bạn muốn. Tôi nhận được một đối tượng - như định dạng, định dạng in JSON cụ thể, với từng bản ghi. Và sau đó là cách viết hoa này ngay đây, và đó là tập hợp các trường này mà tôi phải chọn từ trên xuống khi tôi tạo bất kỳ chuỗi định dạng nào. Một trong những điểm mâu thuẫn là ID, chữ D trong đó, cũng được viết hoa. Vì vậy, một lần nữa, tất cả các lệnh **Docker** đều có sẵn điều này. Nó sẽ rất hữu ích khi bạn bắt đầu làm việc với một cụm và bạn bắt đầu nhận được nhiều loại đối tượng khác nhau và bạn muốn kiểm soát, có thể giới hạn, những gì bạn thấy về một đối tượng nhất định và chỉ tập trung vào những gì bạn quan tâm vì ở đó có thể là một thời gian sau trong khóa học khi chúng tôi thay đổi image, ví dụ: trong việc cập nhật ứng dụng của chúng tôi và chúng tôi sẽ muốn đưa điều đó trở lại sau đó. Có lẽ chúng tôi muốn ném một thứ gì đó khác vào thời điểm đó. Có thể chúng tôi không quan tâm đến ID tại thời điểm đó hoặc có thể tên sẽ không quan trọng với chúng tôi, như trong trường hợp này, bạn có thể thấy tên nhất quán trên tất cả các bản ghi này. Và đó là bởi vì đây là lịch sử của nhiệm vụ mà chúng ta có ở đây. Bạn có thể thấy, ví dụ, nó ở trên m2 ba lần. Đó là ba lần xui xẻo liên tiếp của tôi. Và sau đó là w3, và bây giờ là m3. Vì vậy, đây chỉ là lịch sử nhiệm vụ. Vì vậy, tên thực sự có thể đã được bỏ ở đây. Trong các trường hợp khác, nếu bạn đã mở rộng ứng dụng của mình, bạn có thể có nhiều vị trí, nếu bạn muốn, có nhiều tác vụ để chạy nhiều phiên bản vùng chứa của bạn. Và do đó, cái tên bắt đầu quan trọng vào thời điểm đó và có thể bạn sẽ mang nó trở lại. Vì vậy, hãy coi điều này là bạn cần tập trung vào những gì bạn cần. Chúng tôi có những gì chúng tôi thích ở đây. Tôi sẽ đặt lại lệnh đồng hồ. Hãy bọc nó vào một chiếc đồng hồ, và tôi đặt những câu trích dẫn nhỏ xung quanh vấn đề này. Vì một số lý do, tôi đang gặp sự cố khi định dạng hoạt động khi tôi chuyển nó vào lệnh watch. Chúng tôi đã có bảng của mình và điều tuyệt vời là bảng đó sẽ cập nhật khi chúng tôi thực hiện các thay đổi.

**Docker** Container Kill để Thay đổi Vị trí Quá!

Vì vậy, chỉ để rõ ràng, chúng ta có một biểu diễn dữ liệu tương tự ở đây. Ở phía dưới, chúng tôi có hộp nhỏ này để đại diện cho nhiệm vụ của chúng tôi, nhưng nó chỉ là trường hợp cuối cùng của nhiệm vụ của chúng tôi. Ở trên, chúng tôi thực sự có các phiên bản trước của nó ở định dạng dữ liệu dạng bảng. Vì vậy, khi chúng ta đến đây và làm điều gì đó ở dòng lệnh hơi khác một chút để kích hoạt các thay đổi về vị trí, bạn nghĩ bạn sẽ thấy gì ở trên đây? Được rồi, vì vậy hãy nghĩ về điều đó trong khi chúng tôi làm việc trên phương pháp tiếp cận phụ này để kích hoạt các thay đổi về vị trí, điều này sẽ kích hoạt một câu hỏi khác. Bạn có thể nghĩ về bất kỳ tình huống nào dẫn đến thay đổi vị trí bên cạnh cập nhật bắt buộc mà chúng tôi đã thực hiện không? Vì vậy, một điều tôi thích làm là chỉ để gây ra sự tàn phá. Tôi đã khởi động lại máy ảo của mình và điều đó khiến tác vụ được di chuyển. Thậm chí còn dễ hơn khởi động lại, tôi có thể chuyển ngữ cảnh ở đây, chuyển sang m3, và bạn nghĩ điều gì sẽ xảy ra khi tôi chạy lệnh này, một ps container **docker** và liệt kê mọi thứ? Thực ra, tôi không cần phải liệt kê mọi thứ. Chúng tôi chỉ muốn xem các volume đang chạy. Chúng tôi có thể thấy rằng chúng tôi có vùng chứa của chúng tôi ngay tại đây cho dịch vụ hiển thị image của chúng tôi. Nếu bạn muốn kết hợp mọi thứ, ở cuối tên của container, có u62 này da, da, da, da. Đó là ID nhiệm vụ và nó phải khớp với ID nhiệm vụ ở bên trái ở đây và đúng như vậy. Khi chúng tôi có tất cả những thứ đó tại chỗ, chúng tôi biết mình đang nhắm mục tiêu đúng vùng chứa, chúng ta có thể thực hiện một vùng chứa **docker** và dừng ở đây. Thậm chí có thể mạnh mẽ hơn, chúng tôi có thể giết người, và chúng tôi có thể nhắm mục tiêu viz đó. Sử dụng hoàn thành tab, lười biếng ở đây, chạy nó và hãy xem điều gì sẽ xảy ra. Này, này, bạn có thấy ở trên không? Có thể thấy chúng tôi, hey, chúng tôi đã may mắn. Lần đầu tiên chúng tôi làm điều gì đó gây ra sự tàn phá, nó đã kích hoạt các nhà quản lý đặt nhiệm vụ trên node w2. Và tất nhiên, chúng tôi có thể thử kiểm tra điều đó trong trình duyệt. Vấn đề duy nhất là chúng ta đang có tình trạng suy sụp, điều này thật tuyệt. Và tôi cũng đã khởi động lại máy ảo, vì vậy vùng chứa khác của tôi không hoạt động . Bây giờ, hãy giải quyết những gì đã xảy ra ở đây với dịch vụ của chúng tôi và xem liệu chúng tôi có thể khắc phục sự cố hay không để nó được đặt trên một trong các node quản lý mà nó có vẻ cần.

Cập nhật dịch vụ **Docker** --constraint-add node.role == manager

Vì vậy, chính xác thì thiếu sót ở đây là gì? Chà, có một số cách chúng ta có thể giải quyết vấn đề này. Tôi có thể mở các công cụ dành cho nhà phát triển và xem tab Mạng. Và nếu tôi nhấp xuống đây, nó không phải lúc nào cũng rõ ràng như vậy. Tôi nghĩ đây là một loại thông điệp được thiết kế kém. Nó thực sự sẽ xuất hiện trong giao diện người dùng. Nhưng đây chỉ là một công cụ học tập, nó không phải là một công cụ sản xuất xứng đáng, vì vậy nó ổn. Và này, nếu điều này xảy ra, sẽ không khó để nhận ra rằng bạn sẽ phải di chuyển vùng chứa và thay đổi định nghĩa dịch vụ để nó hoạt động. Vì vậy, những gì nó nói với chúng tôi ở đây, thật khó để nhìn thấy, tôi sẽ mở một tab mới với yêu cầu cụ thể này. Và điều này sẽ dễ đọc hơn một chút. Node này không phải là trình quản lý bầy đàn và không thể sử dụng các node công nhân để xem hoặc sửa đổi trạng thái cụm. Vì vậy, điều này làm nổi bật một phần thông tin quan trọng, đó là các nhà quản lý giữ trạng thái cụm, họ thay đổi trạng thái cụm, họ giữ nguyên trạng thái cụm. Và không có trạng thái nào trong số đó dành cho người lao động ngoài công việc mà họ đang thực hiện. Và có những lý do bảo mật rõ ràng cho điều đó. Chúng tôi cần xác định rằng dịch vụ của chúng tôi chỉ có thể được đặt trên một người quản lý. Để làm điều đó, hãy quay lại định nghĩa dịch vụ của chúng tôi. Chúng ta bắt đầu. Bây giờ ngay bây giờ nó ở dạng một dịch vụ **docker** tạo. May mắn thay, chúng tôi đã thấy rằng có một lệnh cập nhật, vì vậy chúng tôi biết rằng chúng tôi không cần phải xóa lệnh này để sửa chữa nó. Chúng tôi có thể làm điều đó nếu chúng tôi muốn. Nó là một dịch vụ không quốc tịch, vì vậy nó sẽ không làm tổn hại bất cứ điều gì. Nhưng chúng ta hãy xem nó như thế nào nếu chỉ thực hiện một thay đổi nhanh chóng đối với dịch vụ này. Vì vậy, những gì chúng tôi đang tìm kiếm là hạn chế vị trí của dịch vụ của chúng tôi. Vì vậy, chúng tôi sẽ thực hiện cập nhật và chúng tôi cần xác định những gì chúng tôi đang cập nhật. Vì vậy, tôi sẽ cung cấp tên của dịch vụ của chúng tôi ở đây. Và sau đó tôi sẽ đưa ra một dòng ở đây để chúng ta có thể thêm đối số để thực hiện thay đổi mà chúng ta cần thực hiện cho vị trí. Bạn có thể nghĩ về bất kỳ tài nguyên nào chúng tôi có thể sử dụng để tìm ra những gì chúng tôi cần nhập vào đây không? Cũng giống như tạo dịch vụ **docker**, có một tham chiếu tuyệt vời, do đó, là một tham chiếu tốt với cập nhật dịch vụ **docker**. Bạn có thể chỉ cần google cho điều đó với ref và nó thường sẽ kéo bạn đến đúng nơi trong tài liệu dành cho **Docker**, trong phần Tham khảo, cập nhật dịch vụ **docker**. Và đây cũng là một trang thực sự tốt để cuộn qua và đọc vào một thời điểm nào đó. Nó không hoàn toàn lâu như dịch vụ **docker** tạo, nhưng nó có một số thông tin hữu ích bổ sung ở đây. Và nếu tôi cuộn lên trên cùng, tôi có thể quay lại các đối số. Và nếu tôi tìm kiếm vị trí, sẽ có một ưu tiên cho vị trí. Tôi không chỉ muốn một sở thích. Aha, chúng ta bắt đầu, có một hạn chế cũng như vị trí. Và vì vậy trong trường hợp này, chúng tôi sẽ thêm một ràng buộc. Khi xử lý các bản cập nhật, bạn thường nói về việc thêm hoặc xóa một phần cấu hình về trạng thái mong muốn của dịch vụ. Hãy nhớ rằng, các dịch vụ là một đặc điểm kỹ thuật hoặc một trạng thái mong muốn. Chúng mô tả những gì chúng tôi muốn thấy xảy ra, vì vậy chúng tôi cần thêm hoặc xóa vào mô tả đó. Bây giờ trong trường hợp này, chúng ta đang giải quyết một tình huống mà chúng ta có thể sẽ không có thêm thông tin về những ràng buộc nào chúng ta cần bên trong tài liệu này. Bạn có thể nghĩ ra một cái khác mà chúng ta có thể sử dụng không? Vì vậy, có lệnh tạo dịch vụ **docker** đó, và nó rất dài dòng. Nó có rất nhiều thứ hơn cái này. Bên trong đây, chúng ta có thể tìm thấy những thứ chúng ta cần. Vì vậy, tôi sẽ tìm kiếm vị trí từ. Và aha, chúng ta bắt đầu. Đây là một phần tốt đẹp bên dưới. Vì vậy, hãy luôn tìm kiếm các phần bên dưới nếu bạn không chắc chắn những gì cần cung cấp về mặt cấu hình và xem nếu bạn không thể tìm thấy phần mình cần. Bây giờ đây là tùy chọn vị trí một lần nữa, vì vậy đó không phải là những gì chúng tôi cần. Nhìn xung quanh, tôi có thể thấy, không sao, vì vậy thực ra, ở phía bên tay phải , chúng tôi có những ràng buộc về dịch vụ. Đó là những gì chúng tôi đang tìm kiếm. Và do đó, một cách khác để nói về điều này là một hạn chế. Chúng tôi đang hạn chế vị trí, nó làm cho nó trở thành một yêu cầu so với một tùy chọn. Và bây giờ trong trường hợp này, bạn có thể thấy chúng tôi có nhiều thuộc tính khác nhau mà chúng tôi có thể nhắm mục tiêu. Ví dụ, ngay tại đây, chúng ta có thể thấy node.role là trình quản lý. Đó chính xác là những gì chúng ta cần. Vì vậy, tôi có thể sao chép điều đó, tôi có thể liếc xuống bên dưới chỉ để xem cách này hoạt động, vì vậy ‑‑ ràng buộc - thêm, và sau đó đặt vào biểu thức. Và điều đó nên tốt để đi sau đó.

Kiểm tra dịch vụ **Docker**: Cái nhìn đầu tiên về Spec, PreviousSpec và Hội tụ

Được rồi, vì vậy tất cả những gì chúng ta cần làm là nâng lệnh gọi nhỏ của chúng ta ở đây, nó được viết tắt nhiều hơn so với lệnh ban đầu và chúng ta có thể quay trở lại thiết bị đầu cuối và lần này dán lệnh của chúng ta vào. Và khi tôi chạy lệnh này, bạn sẽ làm gì nghĩ sẽ xảy ra ở trên? Hãy cùng tìm hiểu. Được rồi, có vẻ như chúng ta đã có một thay đổi vị trí được kích hoạt, lên lịch lại nhiệm vụ của chúng ta sang một node khác, trong trường hợp này là m2. Bây giờ đó có thể chỉ là may mắn ngẫu nhiên. Một gợi ý rằng mọi thứ vẫn ổn là ứng dụng của chúng tôi sẽ sớm hoạt động. Điều này vẫn không có nghĩa là chúng tôi đã không gặp may, vì vậy tôi sẽ đợi ở đây cho điều đó xảy ra và chúng tôi sẽ quay lại và nói về việc thử nghiệm điều này. Được rồi, mọi thứ đang chạy ngay bây giờ. Dịch vụ của chúng tôi đã hội tụ. Điều đó có nghĩa là nó đã đạt đến trạng thái mong muốn mà chúng tôi đã yêu cầu. Trong trường hợp của chúng tôi, chúng tôi muốn một ứng dụng đang chạy, đó là trạng thái mong muốn ở đây và nó hiện đang chạy. Bạn có thể thấy rằng khi trạng thái hiện tại nhấn đang chạy ở đây, khi nó đang tích tắc, nó đang cập nhật trạng thái hiện tại trong đầu ra của chúng tôi ở đây. Chúng tôi gắn bó với sự hội tụ của dịch vụ để theo dõi quá trình đó xảy ra. Bạn có thể tách ra nếu muốn, nhưng bạn được đính kèm theo mặc định. Và bây giờ dịch vụ của chúng tôi đã hội tụ, điều đó có nghĩa là chúng tôi có thể làm mới cửa sổ này và này, chúng tôi đã có các node. Được rồi. Sẽ có thể làm mới cửa sổ này và chúng tôi có một ứng dụng đang chạy. Vì vậy, đó là tất cả những điều tốt, nhưng điều này không đảm bảo bất cứ điều gì. Có lẽ chúng ta chỉ gặp may. Vì vậy, một số điều chúng ta có thể làm bởi vì chúng ta có thể tiếp tục nhấn một node và kích hoạt thay thế mãi mãi và không bao giờ thực sự chắc chắn rằng những gì chúng ta đang làm đang hoạt động. Một điều khác mà chúng tôi có thể làm, và tôi sẽ cung cấp cho chúng tôi toàn bộ cửa sổ cho việc này vì sẽ có rất nhiều thông tin. Chúng tôi có thể chạy một dịch vụ **docker** và sau đó kiểm tra, và chúng tôi có thể xem xét dịch vụ viz của chúng tôi. Có rất nhiều thông tin sẽ quay trở lại. Chúng tôi sẽ không đi qua hầu hết vấn đề này, nhưng ở đâu đó ở giữa, bạn sẽ thấy một thông số kỹ thuật trước đó và sau đó ở trên đó bạn sẽ thấy một Thông số kỹ thuật hoặc thông số kỹ thuật. Đây là thông số kỹ thuật hiện tại. Vì vậy, hai trạng thái cuối cùng được duy trì. Vì vậy, chúng tôi có thể so sánh điều này để xem liệu ràng buộc của chúng tôi có được thêm vào hay không, nghĩa là nó đã được phân tích cú pháp hay chưa. Nó có vẻ là chính xác. Nó đã được thêm vào đây. Hãy chắc chắn rằng nó không ở bên dưới. Được rồi, vậy nó ở đâu? Hãy đảm bảo rằng nó nằm trong Resources, RestartPolicy, và sau đó là Placement. Vì vậy, hãy xem nếu chúng tôi đạt được điều đó ở đây. Trong PreviousSpec, có Tài nguyên, có Vị trí và không có gì về ràng buộc. Vì vậy, có vẻ như ràng buộc của chúng tôi đã được thêm thành công. Nếu chúng ta muốn, chúng ta có thể khác biệt hai trường riêng biệt này. Điều này mang lại cho chúng tôi một mức độ chắc chắn rằng chúng tôi đã thêm một ràng buộc nào đó và nó dường như đang hoạt động. Tất nhiên, nếu muốn, chúng tôi có thể thử nghiệm và đảm bảo rằng điều này hoạt động lần thứ hai bằng cách buộc một bản cập nhật khác. Được rồi, nó đã di chuyển, nó vẫn ở trên cùng một node và vì vậy chúng tôi có thể làm điều này lặp đi lặp lại và đây sẽ là mức độ chắc chắn khác của chúng tôi. Nhưng điều tốt đẹp là chúng ta có thể thấy trong thông số kỹ thuật cho dịch vụ của mình rằng nó đã có hạn chế đó. Vì vậy, điều đó có nghĩa là nó sẽ dính ở đó trừ khi ai đó gỡ bỏ hoàn toàn hoặc xóa dịch vụ. Và đó là ràng buộc về vị trí này, toàn bộ đặc điểm kỹ thuật này, thông số kỹ thuật hiện tại, điều này được sử dụng để suy ra trạng thái của cụm, trạng thái mong muốn. Và do đó, các node quản lý **Swarm** sẽ đảm bảo rằng trạng thái mong muốn được thực thi. Ví dụ: nếu một node chuyển sang chế độ ngoại tuyến có các vùng chứa đang chạy trên đó là các dịch vụ lưu trữ, thì người quản lý sẽ có trách nhiệm đảm bảo các dịch vụ tương tự chính xác đó được triển khai ở nơi khác, trong trường hợp này là trên node người quản lý.

Thách thức: Triển khai Dịch vụ Web Yêu thích của Bạn ... Tôi sẽ Sử dụng NGINX

Được rồi, vì vậy trình hiển thị của chúng tôi hoạt động tốt vào thời điểm này. Chúng tôi có nó để xác nhận trạng thái của cụm của chúng tôi. Bây giờ tôi muốn thử thách theo cách của bạn. Chúng tôi đã thực hiện rất nhiều trò chơi poker và thúc đẩy bằng dịch vụ hiển thị image của chúng tôi. Chọn một số ứng dụng khác mà bạn biết khá rõ về cách sử dụng image **Docker** cho nó, một cái gì đó giống như một ứng dụng web, có thể là NGINX hoặc Apache và xem liệu bạn có thể triển khai ứng dụng đã nói trên cụm bằng cách sử dụng phương pháp tạo dịch vụ mà chúng tôi đã không được thực hiện cho đến nay. Được rồi, chúng ta hãy cùng nhau điểm qua một trong những điều này. Vì vậy, tôi đã mở trang NGINX tại đây trên **Docker** Hub. Nếu tôi cuộn xuống đây, một nơi nào đó ở giữa sẽ là ví dụ điển hình của bạn về việc chạy ứng dụng và để lộ cổng. Vì vậy, điều này đủ để nhắc nhở chúng ta về những đặc điểm mà chúng ta nên quan tâm. Tôi sẽ làm nổ tung nó một chút, và sau đó chúng ta hãy mở một trình soạn thảo văn bản. Được rồi. Ở phía dưới, tôi có thể gõ các tập lệnh của mình. Vì vậy, tôi bắt đầu với dịch vụ **docker** và tác phẩm của tôi, nhân tiện, Tab + 9 rất tuyệt vời trong những thứ gợi ý sau khi bạn sử dụng nó một thời gian. Hy vọng rằng nó không gợi ý bất cứ điều gì không phù hợp. Được rồi, sau đó chúng ta sẽ đi xuống đây, và sau khi dịch vụ **docker** của chúng ta được tạo, chúng ta cần làm gì tiếp theo? Ở trên đây, chúng ta có một cổng mà chúng ta cần thêm vào, tại sao chúng ta không đưa nó vào? Vì vậy, tôi lười biếng và tôi đã sao chép nó từ dịch vụ khác, có nghĩa là chúng tôi cần phải thay đổi cổng, nếu không chúng tôi sẽ gặp sự cố. Vậy thì sao, chúng ta đi với 8200, được rồi, 8200: 8080, à, đó là cổng 80 bên trong container, cái mà tôi yêu thích khi mọi người thực sự lưu trữ ứng dụng của họ trên cổng 80 bên trong container. Cổng 80 dành cho web, thật tuyệt khi thấy các ứng dụng web thực sự chạy trên cổng đó. Được rồi, và nhìn vào lệnh của chúng tôi ở đây, vì vậy chúng tôi đặt tên cho vùng chứa, điều đó có nghĩa là chúng tôi cần một tên cho dịch vụ của mình. Thay vào đó, chúng ta có một cú pháp tương tự ở đây, vì vậy tên, và sau đó, chúng ta chỉ cần gọi nó là weby. Được rồi, và sau đó chúng ta có một - d về cơ bản là một bộ tách ở đây cũng để chạy như một vùng chứa nền. Chúng tôi có thể tách ra và không chờ đợi sự hội tụ nếu chúng tôi muốn. Đối với điều đó, chúng tôi chỉ xác định cờ tách. Được rồi, bây giờ, chúng ta còn lại gì ở đây? Có vẻ như chúng tôi chỉ cần chỉ định image, đó là tên gợi ý một bản dựng độc quyền của NGINX. Chúng tôi không có điều đó, vì vậy ngay ở phần cuối ở đây, chúng tôi có thể chỉ cần gõ nginx. Và sau đó trước khi tôi quên nó, vì tôi đang viết các dòng ở đây, chỉ để dễ đọc và tối đa hóa khả năng bị đứt đoạn, hãy lấy những dấu gạch chéo ngược đó ở cuối mỗi dòng, được rồi, tốt thôi. Vì vậy, chúng ta hãy yoink này. Và trong một thiết bị đầu cuối, chúng ta hãy tiếp tục và dán cái này vào đây, và xem chúng ta có đúng không. Chéo ngón tay của chúng ta. Được rồi, chúng tôi đã nhận lại được ID cho dịch vụ của mình, vì vậy đó là một tin tốt. Nếu chúng tôi nhận được một số ký tự ngẫu nhiên, tất cả đều tốt. Nhưng tại sao nó không cập nhật ở trên? Đó là một câu hỏi mẹo. Chúng tôi đang theo dõi dịch vụ viz ở đây. Nếu tôi Ctrl + C để hủy điều đó, những gì tôi cần nhập vào đây để chuyển sang theo dõi dịch vụ NGINX, vì vậy, chỉ một thay đổi mà tôi cần thực hiện là chỉ định, này, ps dịch vụ **docker**, và sau đó là tất cả những gì mà gobbledygook cho định dạng, và chúng tôi muốn weby, và bạn cứ việc. Chúng tôi đã có một phiên bản dịch vụ weby của chúng tôi đang hoạt động và nó đã hoạt động được khoảng một phút. Và bạn có thể thấy chúng tôi không có bất kỳ loại thông báo hội tụ nào ở đây ở phía dưới, đó là bởi vì chúng tôi đã chọn tách ra khi chúng tôi tạo ra nó.

Nhật ký dịch vụ **Docker** - Có thể truy cập ngay cả đối với các vùng chứa đang chạy trên các node khác!

Được rồi, chúng ta có thể làm một số việc. Trước hết, có thể sẽ rất tuyệt khi chỉ nhìn thấy giao diện của ứng dụng. Đồng thời khi chúng tôi làm điều đó, tôi muốn cho bạn thấy một tính năng thực sự gọn gàng khác, đó là bạn thực sự có thể xem các tệp nhật ký từ bất kỳ node trình quản lý nào trong cụm và đó là một điều khá mạnh mẽ. Bởi vì bạn nghĩ về nó, volume có thể đang chạy ở bất cứ đâu, và chúng tôi đang kết nối với m3 ngay bây giờ, vì vậy nếu chúng tôi muốn xem các bản ghi, những bản ghi đó phải được chuyển cho chúng tôi để được cung cấp qua m3 bằng cách nào đó. Đó không phải là một kỳ tích dễ dàng. Bạn có muốn đoán cách chúng tôi truy cập vào nhật ký không? Vì vậy, như mọi khi, chúng tôi thực hiện dịch vụ **docker** của mình và nếu bạn nhìn vào đây, bạn sẽ thấy trong đầu ra, chúng tôi có một tùy chọn, hoặc một lệnh con , để xem nhật ký cho dịch vụ của chúng tôi. Vì vậy, chúng tôi chỉ cần nhập nhật ký vào đây, và sau đó chúng tôi cần cung cấp một cái gì đó ở đây. Tôi phải nhập gì? Vì vậy, weby. Và có đầu ra nhật ký của chúng tôi. Nếu có vấn đề gì xảy ra với dịch vụ khởi động, thông tin đó cũng có thể ở đây. Và câu nói đó khiến tôi muốn kiểm tra và xem điều gì đang xảy ra với dịch vụ viz của chúng tôi. Vì vậy, chúng tôi có thể kiểm tra các dịch vụ khác nhau trong khi chúng tôi đang ở đó, và thật không may, nó không có bất cứ điều gì về vấn đề mà chúng tôi đã gặp phải khi chúng tôi triển khai nhầm node.

Lịch sử tác vụ mặc định là 5, Theo dõi nhật ký và Các tùy chọn đầu ra nhật ký khác

Bây giờ theo mặc định, điều này đang hiển thị cho chúng ta mọi thứ có sẵn. Điều đầu tiên tôi muốn quan sát là có năm mục ở đây, ít nhất là năm mục lặp lại. Và nếu bạn nhìn ở phía trước, chúng ta có, đây là cái gì? Chà, nó là một tham chiếu đến vùng chứa của chúng tôi và nó có ID nhiệm vụ bên trong nó, vì vậy về mặt kỹ thuật, nó cũng là một tham chiếu đến nhiệm vụ. Vì vậy, có một trong số họ. Đây là cái thứ hai, thứ ba, thứ tư và thứ năm. Có chính xác năm. Và tôi muốn xác nhận điều đó bởi vì bạn cũng có thể nhận thấy rằng dưới dịch vụ **docker** và sau đó khi chúng tôi nhận ps cho các nhiệm vụ, hãy đặt viz ở đây, bạn có thể nhận thấy có năm mục ở đây. Loại khó đọc với tất cả các thông tin bổ sung, bốn và năm. Tại sao lại có năm và năm? Đó chỉ là sự trùng hợp? Không, nó thực sự đang giữ năm mục. Vì vậy, đó là mặc định là giữ lịch sử tác vụ cho năm tác vụ. Và thật kỳ quặc khi phải đi đâu để cấu hình điều này. Bạn có thể nghĩ rằng nó có thể ở cấp độ dịch vụ. Khi bạn đang thiết lập bầy của mình, lệnh **Swarm** của **docker** đó, khi bạn khởi tạo một bầy, có một loạt các tùy chọn khác nhau mà bạn có thể vượt qua và một trong số đó là ‑‑ task - history - limit. Và nếu bạn để ý ở phía bên tay phải , mặc định ở đây là 5. Vì vậy, đó là lý do xuất phát từ số lượng nhiệm vụ trở lại bạn sẽ giữ cho mỗi dịch vụ và cụ thể là mỗi bản sao hoặc vị trí trong một dịch vụ. Chúng ta sẽ nói nhiều hơn về các bản sao ngay sau đây. Được rồi, và lý do tôi muốn hiển thị nhật ký ban đầu, tôi hơi bị phân tâm, và không sao, có một số điều tốt để xem ở đó, có một số tùy chọn bạn có thể vượt qua và một trong số đó là theo dõi nhật ký, vì vậy hãy làm theo đầu ra nhật ký. Đuôi cũng là một tùy chọn để xem số dòng cuối cùng. Vì vậy, những gì tôi sẽ làm là tôi sẽ chạy lại phần này, đưa vào ‑‑ theo dõi ở đây và chúng tôi cần cung cấp weby. Nếu bạn nhìn vào cuối màn hình, bạn sẽ nhận thấy rằng tôi vẫn chưa nhận lại được lời nhắc của mình và đó không chỉ giống như một sự tình cờ. Nếu tôi nhấn Return, nó vẫn không quay trở lại. Đó là bởi vì tôi thực sự đang lắng nghe ở đây cho bất kỳ đầu ra tiếp theo nào.

**Swarm** Networking Magic - Truy cập các cổng đã xuất bản từ bất kỳ node nào!

Vì vậy, hãy giữ đầu ra nhật ký đó mở trong khi chúng ta truy cập trình duyệt ở đây và tải lên ứng dụng NGINX của chúng ta. Tôi cần nhập gì vào đây để truy cập ứng dụng NGINX đó? Vì vậy, đây là lúc một chút kỳ diệu xuất hiện. Tất cả những gì bạn phải làm là đánh vào một trong các node trong cụm. Bạn không cần phải quan tâm xem bạn đã đánh vào node nào. Vì vậy, chúng tôi đã sử dụng 102 trước đây. Hãy sử dụng cái đó một lần nữa. Và sau đó, bạn có nhớ cảng không? Đó là 8200. Này, này, nhìn này. Chúng tôi đã có ứng dụng của mình và điều đó khá tuyệt. Điều thú vị là tôi có thể đến đây và thay đổi điều này thành 3, và nó sẽ tải. Và này, hãy nhìn vào phía dưới. Bạn có thể thấy đầu ra nhật ký từ các yêu cầu đến, đó là những gì tôi đã hy vọng. Có 4. Trên thực tế, không có 4. Chúng tôi không có người quản lý thứ tư. Mặc dù vậy, chúng tôi có một công nhân 4. Nhìn kìa. Nó được tải trong trường hợp của một công nhân. Hãy đến với worker 2, 112. Vì vậy, từ góc độ bên ngoài, nếu bạn tò mò về cách bạn truy cập vào các dịch vụ đang chạy trên cụm của mình, bên ngoài cụm của bạn, điều đó thực sự dễ dàng. Bạn có thể nhấn bất kỳ node nào trong số các node và nếu bạn có một cổng đã xuất bản và từ xuất bản ở đây, thì từ đó sẽ có sẵn cho tất cả các node bên ngoài trên cổng đó.

Loại bỏ các dịch vụ và theo dõi việc loại bỏ với các sự kiện có quy mô bầy đàn

Tôi muốn xem xét một cách tiếp cận khác về cách chúng ta có thể theo dõi cụm của mình từ dòng lệnh và đó là sử dụng lệnh sự kiện **docker**. Vì vậy, nếu bạn chỉ chạy điều này trên một node người quản lý, ở đó bạn sẽ có quyền truy cập vào những gì chúng ta sẽ thấy là cả sự kiện bầy đàn và sự kiện cục bộ. Vì vậy, bạn có thể thấy một số sự kiện đã xuất hiện ở đây, tôi sẽ làm rõ những điều này, chúng ta sẽ nói về những sự kiện đó trong một phút. Những gì tôi muốn làm là chia đôi màn hình và kích hoạt một trong những sự kiện của riêng chúng tôi. Và ngay bây giờ, nếu tôi xem xét các dịch vụ của mình, nhìn vào dịch vụ weby, bạn nghĩ tôi sẽ làm gì để xóa dịch vụ đó, và bạn nghĩ điều gì sẽ xảy ra ở trên cùng khi tôi làm điều đó? Vì vậy, dịch vụ **docker**s rm, và sau đó để kích hoạt điều này, tôi sẽ cần chỉ định dịch vụ nào tôi muốn loại bỏ và tôi sẽ loại bỏ weby trong trường hợp này. Chúng tôi sẽ có một số sự kiện hiển thị liên quan đến việc dịch vụ của chúng tôi bị xóa. Vì vậy, điều này hơi lộn xộn vì gói, đó là một điều tôi không thực sự khuyến khích khi xem xét các sự kiện, vì vậy tuy nhiên bạn cần phải làm điều đó để thoát khỏi gói, tôi có thể sử dụng lệnh less. Và sau đó tôi sẽ quay lại lệnh sự kiện ở đây, tôi sẽ quay ngược thời gian có thể là 2 phút trước. Vì vậy, bây giờ bạn có thể thấy đầu ra nhật ký kết thúc bên ngoài màn hình ở đây, và sau đó ở giữa, chúng ta có thể thấy tổng quan cấp cao về các sự kiện đã xảy ra. Trong trường hợp này, chúng tôi có một số lệnh thực thi đã xảy ra. Đây là những gì kiểm tra sức khỏe. Vì vậy, dịch vụ viz, trình hiển thị, có các kiểm tra sức khỏe được tích hợp sẵn và đây chỉ là trạng thái của những trình hiển thị đó và tôi tin rằng đó là bởi vì trình hiển thị đang chạy trên node cụ thể này. Và có vẻ như 2 phút quay lại là không đủ xa, vì vậy tôi sẽ quay lại có thể là 5 phút vào thời điểm này. Chúng ta bắt đầu, đó là những gì tôi muốn ngay tại đây. Đây là dịch vụ của chúng tôi loại bỏ. Một cách khác để chúng ta có thể nhắm mục tiêu một loại sự kiện cụ thể là đi vào lệnh sự kiện và ngoài ra, chúng ta có thể thêm bộ lọc và sau đó nếu bạn hoàn thành Tab, tôi thấy đó là cách tốt nhất để truy cập các bộ lọc khác nhau. có thể muốn thêm. Cũng giúp với cú pháp ở đây. Vì vậy, bạn có thể hoàn thành Tab thông qua đây và những gì chúng tôi đang tìm kiếm là loại sự kiện và đối với loại, nếu tôi nhấn lại Tab, tôi biết rằng tôi đang xem các sự kiện cấp dịch vụ , vì vậy tôi sẽ đặt điều đó trong. Khi tôi làm điều đó, bạn có thể thấy bây giờ chúng tôi vừa có sự kiện dịch vụ quay trở lại. Một cách khác tôi có thể cắt và xúc xắc, thay vì bộ lọc kiểu, tôi có thể sử dụng bộ lọc phạm vi. Nếu tôi quay lại đây, tôi có thể hiển thị lại cho bạn tất cả các bộ lọc đó. Vì vậy, chúng tôi có phạm vi, khá nhiều chúng tôi có bộ lọc trên bất kỳ thứ nguyên tài nguyên nào mà chúng tôi có thể quan tâm. Và đối với bộ lọc phạm vi đó một lần nữa, tôi không thể hoàn thành bất kỳ nội dung nào ở đây, nhưng tôi biết rằng tôi có sẵn cục bộ và bầy đàn, và có vẻ như 5 phút trở lại bây giờ là không đủ dài. Hãy quay lại 9 phút, chúng ta bắt đầu. Chúng tôi đã xóa dịch vụ của mình. Vì vậy, đây sẽ là tất cả các sự kiện bầy đàn của chúng tôi trong 9 phút qua. Và đó là điều gọn gàng, nếu bạn thực sự muốn quay ngược thời gian, bạn có thể. Tôi muốn nói rằng có thể là 109 phút, bạn sẽ thấy mọi thứ tôi đã làm và bất kỳ sự chuẩn bị nào cho bất kỳ cuộc biểu tình nào ở đây trong một giờ qua - và - một - nửa hoặc lâu hơn. Ví dụ: bản cập nhật node ở đây, đó là vì cụm của tôi ngoại tuyến, cũng như bản cập nhật mạng. Chúng ta sẽ nói về mạng ngay sau đây. Đây chỉ là những thứ trực tuyến trở lại khi tôi bật cụm của mình, máy của tôi đã chuyển sang chế độ ngủ và vì vậy tôi phải khởi động lại mọi thứ bằng Vagrant. Nhật ký này - giống như đầu ra, nó khá hữu ích, nhưng nó chứa rất nhiều thông tin và rất dễ dàng để nó xuất hiện trên màn hình, bạn phải giải quyết vấn đề đó hoặc có thể bạn phải thu phóng theo cách, cách, cách, cách ra và thử và lấy tất cả thông tin trên màn hình, đôi khi có thể hoạt động tùy thuộc vào loại sự kiện diễn ra.

Lịch sử sự kiện **Docker** có thể đọc được với JSON và jq

Giống như tất cả các lệnh liệt kê và theo đuôi khác mà chúng tôi đang xem xét đầu ra ở một số loại định dạng cuộn, thường có sẵn một tùy chọn định dạng và nó thường có một số loại mẫu Go. Vì vậy, giống như trước đây, chúng ta có thể thiết lập mẫu này bằng JSON, lấy toàn bộ ngữ cảnh và in nó ra từng đối tượng và chúng ta sẽ có được đầu ra JSON đẹp mắt này. Thật không may, nó không được định dạng. Tuy nhiên, chúng tôi có thể làm điều gì đó về điều đó. Tôi muốn chuyển điều này đến lệnh jq. Nó không chỉ định dạng và thụt lề cho JSON, vì vậy nó rất dễ đọc, trong đó có nhiều hơn hoặc ít hơn một cặp khóa - giá trị trên mỗi dòng, nếu bạn thành thạo sử dụng jq, bạn có thể sử dụng nó để lọc đầu ra và thay đổi đầu ra. bạn muốn. Vì vậy, điều tốt đẹp là nếu tôi quay ngược thời gian xa hơn một chút ở đây, hãy quay lại 30 phút, và quay lại và thả bộ lọc, chúng tôi bắt đầu, chúng tôi nhận được các sự kiện vùng chứa của chúng tôi và mọi thứ khác. Vì vậy, chúng tôi có rất nhiều thông tin phong phú để xem nếu bạn tò mò về tất cả những gì có sẵn. Thật là một cách tuyệt vời để có được ý tưởng về những gì có thể bằng cách quay ngược thời gian. Và đây là nơi bạn có thể muốn lọc các loại sự kiện khác nhau. Ví dụ: execute\_create, execute\_start. Đó là cả hai bước khởi đầu và kết thúc của việc kiểm tra sức khỏe. Vì vậy, có một lệnh thực thi chạy để truy cập vào vùng chứa. Sau khi hoàn tất, sẽ có một tệp thực thi. Vì vậy, sau khi kiểm tra sức khỏe hoàn tất, nó sẽ có thông tin trở lại, và điều này thể hiện thông tin đó sẽ quay trở lại. Bây giờ, đây không phải lúc nào cũng có trong một mô hình mà bạn có được một loạt các chi tiết về sự kiện. Bạn nhận được mã thoát, trong trường hợp này, thật tuyệt. Tuy nhiên, không phải lúc nào bạn cũng nắm được tất cả các chi tiết. Ví dụ: khi chúng tôi xóa dịch vụ của mình, chúng tôi không nhận được bất cứ điều gì về các tác vụ đã bị xóa hoặc các vùng chứa đã bị xóa hoặc quá trình xảy ra sự kiện đó, vì vậy một số sự kiện phong phú hơn những sự kiện khác có thể là do rất nhiều sự kiện nhóm này đã được thêm gần đây hơn các sự kiện khác. Trên thực tế, nếu bạn muốn xem tất cả những gì có sẵn, nếu bạn đi ra bến tàu sự kiện **Docker** và cuộn xuống dưới cùng, bạn sẽ bắt đầu thấy các ví dụ, chẳng hạn như sự kiện node, loại sự kiện bí mật, cấu hình, sau đó đây là dịch vụ. Vì vậy, đối với dịch vụ, chúng ta sẽ chỉ thấy việc tạo và xóa. Đối với các plugin, chúng ta sẽ thấy việc kéo và bật các plugin đó. Vì vậy, bạn có thể truy cập vào đây và xem tất cả những gì có sẵn và một số sự kiện có sẵn nhiều thông tin hơn về chúng. Những thứ này có thể tốt chỉ để xem những gì đang diễn ra và học hỏi, hoặc thậm chí trong môi trường sản xuất, nếu bạn muốn theo dõi những thứ đang diễn ra, thì bạn có thể phản ứng và truy vấn trạng thái của cụm để tìm ra bức tranh chi tiết về những thứ nếu bất kỳ sự kiện nào trong số này không có tất cả các chi tiết bạn muốn.

Triển khai ứng dụng với ngăn xếp

Một ngăn xếp để triển khai dịch vụ NGINX weby của chúng tôi

Trong mô-đun cuối cùng, chúng tôi đã xem xét khái niệm mới này được gọi là dịch vụ giúp chúng tôi triển khai các vùng chứa trên một cụm máy, cụ thể là một bầy. Trong mô-đun này, chúng ta sẽ xem xét bước tiếp theo, nếu bạn muốn, trong quá trình phát triển cách chúng tôi triển khai các ứng dụng của mình vào một cụm bằng cách xem xét các ngăn xếp. Và nếu bạn có một phút, có thể bạn dành một giây, tạm dừng video và nghĩ về chính xác những gì một ngăn xếp có thể đại diện cho điều mà bạn có thể đã quen thuộc. Vào cuối mô-đun cuối cùng, chúng tôi có hai dịch vụ, viz và weby. Chúng tôi đã yêu cầu weby xem xét các sự kiện **Docker** và xem xét quá trình xóa một dịch vụ. Vì vậy, tại sao chúng ta không mang weby trở lại lần này với một ngăn xếp? Trong Visual Studio Code, một vài điều. Tôi đã tạo một thư mục ví dụ và bên trong đó tôi có một thư mục dịch vụ. Tôi thiết lập nó và sao chép các tập lệnh này để tạo các dịch vụ khác nhau từ mô-đun cuối cùng. Đối với mô-đun này, hãy mở thư mục ngăn xếp. Bên trong đây, tôi có nhiều ví dụ khác nhau và mở ra cụ thể bộ ví dụ weby. Có ba tệp ở đây. Chúng tôi có weby như một vùng chứa, như một dịch vụ và chúng tôi có weby như một ngăn xếp. Tệp ngăn xếp này trông có quen thuộc không? Điều này sẽ nhắc nhở bạn về tệp Soạn thư, cụ thể là tệp **Docker** Compose, vì nguyên văn nó sẽ hoạt động bên trong **Docker** Compose để triển khai như một dự án hoặc ứng dụng độc lập và nó cũng sẽ hoạt động ở đây như một ngăn xếp. Và đó là điểm tương đồng quan trọng với các ngăn xếp sẽ giúp bạn tiến về phía trước và có thể học cách yêu thích chúng rất nhiều. Trước khi bạn biết điều đó, bạn sẽ từ bỏ lệnh tạo dịch vụ đó bởi vì việc gõ các đối số ra ngoài thực sự không thú vị, rất giống với việc chạy vùng chứa của **docker** so với do trình soạn thảo của **docker**. Vì vậy, điều đầu tiên, chúng ta hãy xem xét weby - container, và hãy so sánh nó với weby - service để xem hai tệp này giống nhau như thế nào. Được rồi, và tôi biết nếu tôi vào Cài đặt ở đây, tìm điểm khác biệt và thêm khoảng trắng, tôi sẽ chọn hộp này ở đây để bỏ qua khoảng trắng ở đầu và cuối, và bạn có thể thấy sự khác biệt dường như thậm chí còn có nhiều điểm chung . Ngoài ra, chúng ta chỉ có hai từ khác nhau. Vì vậy, bây giờ chúng ta chỉ cần lấy dịch vụ từ dịch vụ **docker** tạo ở phía bên phải, và hãy so sánh nó với weby - stack. Nó sẽ không phải là một quả táo để so sánh táo. Có đủ sự khác biệt để các đường kẻ đều có màu, nhưng phần quan trọng là bất kỳ vùng màu tối nào, ví dụ như màu đỏ sẫm hoặc xanh lá cây đậm, với 8200 được ánh xạ đến 80 ở đây, điều tương tự ở đây, bóng tối hơn cho biết văn bản phù hợp và nằm trên một dòng có văn bản không khớp. Ví dụ: bản ‑‑ thì không, và do đó, ở đây có màu đỏ nhạt hơn. Nếu bạn nhìn vào thứ có màu đỏ nhạt, chúng ta có thể thấy một số điểm trùng lặp. Chúng tôi đang tạo một dịch vụ ở đây và chúng tôi đang lập bản đồ cổng này. Bây giờ chúng ta đã có tên và image được lật lại trong hai ví dụ. Vì vậy, những gì tôi có thể làm là tạm thời xem qua và di chuyển image xuống để image xuất hiện sau đó, và bây giờ bạn có thể thấy tất cả các đối số của chúng tôi, vì vậy những gì chúng tôi đang tạo, một dịch vụ, với tên weby với cổng 8200 được ánh xạ thành 80 và sau đó với image NGINX. Ngoài những thông số này, nếu bạn muốn, phần còn lại chỉ là giàn giáo để thực hiện yêu cầu của chúng tôi. Trong một trường hợp, chúng tôi sử dụng tạo dịch vụ **docker** và trong trường hợp khác, chúng tôi sử dụng ngăn xếp, mà vào cuối ngày, thực sự cũng sẽ tạo ra một dịch vụ. Như bạn có thể thấy, chúng tôi có một khối dịch vụ ở đây. Vì vậy, thật đáng ngạc nhiên là có bao nhiêu vẫn khớp ngay cả khi chúng ta đang chuyển từ tập lệnh shell ở bên trái sang YAML ở bên phải. Nhân tiện, tôi đã liên kết với một tài liệu lược đồ Soạn đặc tả và đó là lý do tại sao bạn nhìn thấy di chuột qua đó cho biết soạn thảo - đặc điểm một lúc trước. Phần này lên đây là hoàn toàn không cần thiết. Nó là tùy chọn. Nó liên quan đến plugin Red Hat tích hợp sẵn mà tôi đang sử dụng trong VS Code về máy chủ ngôn ngữ YAML để xác định lược đồ cho tài liệu mà tôi đang làm việc. Vì vậy, bỏ qua điều đó và bỏ qua shebang ở phía bên tay trái , tôi muốn nói rằng hai tệp này sẽ trông quen thuộc nếu bạn thậm chí chưa bao giờ làm việc với **Docker** Compose

Triển khai ngăn xếp weby với Triển khai ngăn xếp **Docker**

Được rồi, vậy khi chúng ta bắt đầu triển khai ngăn xếp của mình, bạn nghĩ điều gì sẽ khác ở đây ở dòng lệnh? Vâng, trước tiên, chúng ta đã có trước điểm nhập dịch vụ **docker**, một số lệnh quản lý với một loạt lệnh con để giúp chúng ta quản lý từng dịch vụ mà chúng ta đang tạo và cập nhật, xóa, v.v. trong mô-đun cuối cùng. Trong mô-đun này, chúng ta đang nói về ngăn xếp. Và theo đó, có một lệnh quản lý ngăn xếp **docker** rất hay có một loạt các lệnh con. Một số người trong số họ trông rất quen thuộc. Ví dụ, chúng tôi có ls và ps. Vì vậy, chúng tôi chỉ có thể chạy cái này, ls ngăn xếp **docker**. Bạn sẽ thấy chúng tôi không có bất kỳ ngăn xếp nào. Rõ ràng, chúng tôi đã không triển khai bất kỳ. Và đối với ps ngăn xếp **docker**, chúng ta sẽ cần triển khai một ngăn xếp để có thể xem xét các nhiệm vụ của nó. Vì vậy, tôi đang ngồi bên trong thư mục weby, và có tệp ngăn xếp của tôi. Để triển khai điều này, tất cả những gì tôi phải làm là ngăn xếp **docker** và sau đó triển khai, chỉ định tệp ngăn xếp với - c arg. Và sau đó tôi cũng cần đặt tên cho nó, vì vậy chúng tôi sẽ gọi nó là ngăn xếp weby . Ngay lập tức, chúng tôi được trình bày với việc tạo ra một mạng và một dịch vụ. Vậy được rồi, một mạng, điều đó thật kỳ quặc. Chúng tôi không yêu cầu điều đó, tôi không nghĩ vậy. Nhưng cũng giống như với **Docker** Compose, có các tính năng ngầm và mạng là một trong những tính năng ngầm đó, và đó là lý do tại sao chúng tôi nhận được mạng weby\_default. Thậm chí có thể nhìn vào ls mạng **docker**, và bạn có thể thấy chúng tôi đã tạo mạng weby\_default và lần này, thật thú vị, nó thực sự là một trình điều khiển lớp phủ. Nếu bạn đã sử dụng **Docker** Compose trước đây, bạn sẽ thấy một trình điều khiển cầu nối trên mạng do người dùng của bạn xác định. Đây là lớp phủ bởi vì chúng tôi đang xử lý một cụm máy, bạn có thể thấy phạm vi là một đám, vì vậy nó phải mở rộng ra ngoài việc chỉ kết nối các thiết bị với nhau trên một thiết bị duy nhất. Vì vậy, đó là cách vùng chứa của chúng tôi có được kết nối. Nhưng điều quan trọng nhất là dịch vụ của chúng tôi đã được triển khai, vì vậy chúng ta hãy xem lại lệnh **docker** stack ls. Bạn có thể thấy chúng tôi có hệ thống weby của chúng tôi và đang chạy với một dịch vụ. Bây giờ chúng ta có thể xem xét ngăn xếp **docker** và sau đó ps nếu chúng ta muốn, kết xuất các tác vụ cho ngăn xếp weby của chúng tôi, tất cả chúng và có một tác vụ đại diện cho một vùng chứa duy nhất đang chạy như một phần của weby của chúng tôi cây rơm. Nếu tôi muốn mở nó trong một trình duyệt, bạn nghĩ tôi cần nhập gì vào thanh URL? Vì vậy, giống như trước đây, tôi cần nhập địa chỉ IP của bất kỳ node nào trong cụm và sau đó chúng tôi đang chạy trên cổng 8200 là cổng mà chúng tôi đã xuất bản. Và có dịch vụ NGINX của chúng tôi. Vì vậy, nó được thiết lập và chạy. Điều đó khá tuyệt. Nếu chúng ta muốn xem xét thêm một số điều, chúng ta có thể xem lệnh **docker** service ls, và bạn có thể thấy chúng ta có cả viz của mình và lần này chúng ta có weby\_weby cho dịch vụ của mình. Và đó là bởi vì, giống như với các dự án **Docker** Compose, chúng tôi đặt tiền tố cho tên ngăn xếp, vì vậy weby là tên chúng tôi chọn cho ngăn xếp, trước các tài nguyên khác nhau bên trong ngăn xếp đó, chứa dịch vụ weby của chúng tôi. Và tất nhiên, chỉ muốn trong mô-đun cuối cùng, nếu chúng tôi có thể xem xét các nhiệm vụ của mình theo cách này, nhưng chúng tôi phải cung cấp weby\_weby. Đó là lý do tại sao hoàn thành tab là rất tốt để thiết lập. Và có thông tin tương tự như trên khi chúng ta sử dụng ps ngăn xếp **docker**. Vì vậy, có rất nhiều cách khác nhau để có được cùng một thông tin để xem điều gì đang xảy ra. Vào cuối ngày, tôi hy vọng bạn sẽ thấy thuận tiện hơn khi làm việc với một tệp về việc truyền các đối số, do đó bạn không phải sao chép và dán các lệnh và bạn không phải lo lắng về việc nhập các đối số bằng tay. Một tệp có thể được kiểm tra trong kiểm soát phiên bản. Đó là một điều tuyệt vời khi được giấu kín một cách an toàn với một lịch sử thay đổi tốt đẹp.

Cập nhật dịch vụ đã triển khai - Chỉ cần chỉnh sửa tệp ngăn xếp và nhấn "node triển khai" một lần nữa!

Bây giờ là sức mạnh thực sự của tệp ngăn xếp, cũng như sức mạnh thực sự của tệp Soạn. Giả sử chúng tôi muốn thay đổi điều gì đó về dịch vụ weby đã triển khai của chúng tôi, đó là một phần của ngăn xếp weby của chúng tôi . Ví dụ: làm thế nào về việc thay đổi cổng mà chúng tôi đã xuất bản? Hãy chuyển từ 8200 sang 8300. Bạn nghĩ chúng tôi làm gì để thiết lập điều đó? Vì vậy, tất cả những gì chúng ta phải làm là chỉnh sửa tệp này ngay tại chỗ. Đặt 8300 ngay tại đây. Quên đi giá trị cũ. Lưu các tập tin. Nhảy ngay sang dòng lệnh. Tôi có thể chạy một dịch vụ **docker** nhanh để chúng tôi có thể xem xét cổng mà chúng tôi đang chạy. Bạn có thể thấy chúng tôi đang sử dụng 8200 ngay bây giờ cho dịch vụ weby\_weby của mình. Bây giờ bạn nghĩ tôi sẽ làm gì để triển khai các thay đổi mà tôi đã thực hiện đối với tệp đó? Tôi chỉ cần chạy lại lệnh triển khai. Tôi sẽ loại bỏ điều đó bằng cách sử dụng cùng một tên cho ngăn xếp của chúng tôi. Hãy nhớ rằng bạn có thể tạo các phiên bản riêng biệt của cùng một tệp ngăn xếp, vì vậy bạn sẽ muốn giữ nguyên các tên trừ khi bạn có ý định tạo một phiên bản riêng biệt. Và đó là điều thú vị khi sử dụng các tệp ngăn xếp, gần giống như một hàm mà bạn có thể chèn các đối số mới và lấy một thể hiện mới, một mẫu, nếu bạn muốn. Được rồi, bạn thấy nó thông báo Đang cập nhật dịch vụ weby\_weby? Hãy chạy ls dịch vụ **docker** của chúng tôi, nhưng bạn có muốn đoán những gì sẽ hiển thị trong đầu ra ở đây cho các cổng không? Chà, này, nhìn kìa. Bây giờ chúng tôi có 8300 thay vì 8200. Nhưng thử nghiệm thực sự là chuyển sang trình duyệt của chúng tôi, đảm bảo rằng 8200 đã bị hỏng. Thật tuyệt. Hãy luôn yêu thích nó khi chúng ta đang tìm kiếm những điều bị phá vỡ. Và 8300 đang hoạt động. Đó là một cách cực kỳ dễ dàng để thực hiện thay đổi đối với một ứng dụng đã triển khai để thay đổi cổng mà chúng tôi đang chạy. Nó không phải lúc nào cũng là một điều dễ dàng để thực hiện. Điều đó yêu cầu loại bỏ cổng đã xuất bản khác và sau đó thêm vào một cổng đã xuất bản mới. Nó cũng yêu cầu kiểm tra trạng thái hiện tại của mọi thứ để xem liệu chúng tôi có bất kỳ cổng nào được xuất bản hay không và sau đó làm thế nào để loại bỏ chúng và sau đó làm thế nào để thêm một cổng mà chúng tôi cần. Tất cả đều được gói gọn cho chúng ta về logic bên trong **Docker** **Swarm**, cụ thể là logic **Swarm**kit, là nơi điều này cư trú. Đảm nhận việc xử lý rất nhiều tình huống điên rồ cho chúng tôi để chúng tôi có thể chỉnh sửa một tệp văn bản vào cuối ngày, nhấn một node và chúng tôi đã hoàn tất. Đó là một quy trình làm việc rất thuận tiện. Vì vậy, nếu bạn đã quen với **Docker** Compose, điều này sẽ cảm thấy như ở nhà. Nó thậm chí còn mạnh hơn **Docker** Compose vì bạn thực sự có một cụm với quy trình điều chỉnh đang chạy cho bạn. Bạn không cần phải chạy **Docker** Compose để điều chỉnh diễn ra một lần nữa. Nó luôn xảy ra cho bạn. Và cách tốt nhất để hiểu những gì tôi vừa nói là xem một ví dụ về việc thay đổi các node trên cụm của chúng tôi sao cho vùng chứa mà chúng ta đang chạy không còn chạy ở vị trí của nó nữa vì có thể chúng ta đã rút phích cắm trên một node và xem điều gì sẽ xảy ra sau đó với vùng chứa weby của chúng tôi, đây là một phần của ngăn xếp weby của chúng tôi .

Làm thế nào để giám sát tất cả các nhiệm vụ tại CLI

Ví dụ tiếp theo này sẽ yêu cầu sửa đổi một trong các node của chúng ta sao cho một trong các nhiệm vụ của chúng ta đối với một dịch vụ phải chuyển đi nơi khác. Và tôi chỉ nhìn vào điều này và nghĩ rằng, chúng tôi đang làm việc với dịch vụ weby ngay bây giờ. Tại sao chúng ta không làm gì đó với m2 để weby phải dọn đi? Và tôi đã nghĩ rằng thực sự có một trường hợp sử dụng thực tế ở đây. Tôi rất vui vì weby đã hạ cánh trên m2. Trường hợp sử dụng đó là các node người quản lý có xu hướng chỉ cần là các node người quản lý, đặc biệt là khi bạn phát triển cụm của mình. Họ cần tiết kiệm tài nguyên cho nhiệm vụ trở thành người lãnh đạo hoặc về cơ bản là người theo dõi và tái tạo trạng thái của cụm trong trường hợp xảy ra một số loại kịch bản chuyển đổi dự phòng để họ có thể thăng tiến và trở thành người dẫn đầu. Vì vậy, giả sử rằng chúng tôi muốn thay đổi m2 để hoàn thành nhiệm vụ đó, để cung cấp tất cả các nguồn lực của chúng tôi trên m2 cho nhiệm vụ quản lý. Chà, chúng ta có thể thực hiện cái gọi là rút bớt một node để biến một node người quản lý chỉ một node người quản lý một cách hiệu quả vì theo mặc định, chúng có thể vừa là người quản lý vừa là người làm việc. Bây giờ để làm điều đó, tôi đã thiết lập một tập lệnh mà chúng ta có thể sử dụng. Điều này sẽ giúp chúng tôi theo dõi mọi thứ khi chúng tôi thực hiện thay đổi. Đó là màn hình này - các node\_tasks bên trong thư mục máy chủ bên trong thư mục autos. Bây giờ điều này không xảy ra với ngăn xếp weby của chúng tôi , và đó là bởi vì nó cần được truy cập dễ dàng từ máy chủ **Docker** của chúng tôi. Và bởi vì tôi đang sử dụng Vagrant VM, rất tình cờ là có một thư mục chia sẻ được thiết lập bắt nguồn từ Vagrantfile này và vì vậy thư mục máy chủ này nằm bên trong đó sẽ cho phép chúng tôi dễ dàng truy cập vào các tệp khác bên trong đó thư mục chia sẻ. Chúng tôi sẽ thấy điều đó trong một phút. Vì vậy, màn hình hiển thị này - các node\_tasks, bên trong đây tôi có hai lệnh, thực tế là ba lệnh. Tôi có một node **docker** ở trên để liệt kê các node của chúng tôi và sau đó tôi có một chuỗi định dạng tùy chỉnh ở đây. Sau đó, ở bên dưới, tôi có tất cả các tác vụ và nó in ra thông tin về tất cả các tác vụ và nó thực hiện điều đó với ps dịch vụ **docker** và được nhúng bên trong nó là một cuộc gọi đến dịch vụ **docker** ls - q. Và dịch vụ **docker** ls - q, chúng ta có thể chưa thấy điều đó, vì vậy chúng ta hãy xem xét điều đó. Bạn nghĩ điều đó sẽ ra sao nếu tôi đang nhúng nó vào cuộc gọi khác? Hãy chạy nó và tìm hiểu. Chúng tôi chạy cái này, đây là hai mục hàng riêng biệt và đại diện cho hai dịch vụ riêng biệt mà chúng tôi có. Bạn có thể thấy ID của họ khớp với nhau. Điều thú vị là nếu bạn chỉ in ra các ID, chúng có thể được đưa vào lệnh ps dịch vụ **docker**, mà chúng tôi biết yêu cầu một hoặc nhiều dịch vụ. Vì vậy, chúng tôi đang cung cấp tất cả các ID dịch vụ của mình và nhận tất cả các nhiệm vụ dịch vụ của chúng tôi vào thời điểm đó. Bạn có thể thấy nó trông như thế nào bằng cách chạy tập lệnh trong vagrant / host / display - node\_tasks. Và ở đó bạn đi. Bạn có thể xem tất cả thông tin được in ở trên. Bây giờ tôi đã chia màn hình để chúng ta có thể chạy các lệnh bên dưới vì những gì tôi muốn làm là xem tập lệnh này ở trên để tôi có thể nhận thấy những thay đổi là gì khi chúng tôi thực hiện thay đổi. Vì vậy, đây là một cách khác để theo dõi trạng thái của mọi thứ, là xây dựng một tập lệnh hiển thị nhỏ, nếu bạn muốn, với bất cứ thứ gì bạn muốn bên trong nó và sau đó xem tập lệnh hiển thị đó. Được rồi, vậy là tôi đã lưu lệnh ở đây. Lưu ý rằng chúng tôi chỉ đi qua để xem chính xác con đường mà tôi vừa chạy riêng lẻ. Vì vậy, cứ mỗi nửa giây, chúng tôi sẽ thực hiện điều đó, và đó là lý do tại sao tôi đang ngồi ngay trên máy chủ **Docker** của mình bên trong máy ảo Vagrant. Tôi muốn có màn trình diễn để có thể nâng tầm nó lên đến khoảng thời gian này. Bên ngoài máy chủ lưu trữ **Docker** của bạn trên máy khách, có thể muốn chuyển điều này thành 1 đến 2 giây, tùy thuộc vào thời gian tập lệnh này chạy trong bao lâu. Được rồi, vì vậy tôi chạy nó, và bây giờ nó đang cập nhật mỗi nửa giây, đại khái.

Hiểu đối chiếu bằng cách thoát node quản lý

Được rồi, bây giờ ở bảng dưới cùng, tôi có thể chạy các lệnh và chúng ta có thể xem những gì xảy ra ở trên. Vì vậy, tôi sẽ nhanh chóng tạo một tab mới để chúng ta có thể xem xét lệnh của node **docker**. Vì vậy, chúng tôi có một vài lệnh mà thoạt đầu có thể giống như những gì chúng tôi cần. Có một thăng chức, và có một hạ cấp. Và chúng tôi có thể điều hành việc này trên người quản lý, nhưng sau đó chúng tôi đang biến người quản lý chỉ là một công nhân nếu chúng tôi cách chức hoặc trong trường hợp một công nhân, nếu chúng tôi thăng chức, chúng tôi sẽ biến người quản lý thành một người quản lý, sau đó. Chúng tôi không muốn điều đó. Thay vào đó, chúng tôi muốn câu lệnh con cập nhật này, vì vậy hãy tìm một số trợ giúp cho việc đó và bắt đầu. Chúng tôi có những gì chúng tôi cần ở đây. Chúng tôi đang tìm kiếm sự sẵn có. Bạn có thể thấy ở phía bên tay phải , chúng tôi có ba lựa chọn cho tính khả dụng, hoạt động, tạm dừng và tiêu hao. Hoạt động là nơi chúng tôi đang ở ngay bây giờ. Tạm dừng sẽ không thêm bất kỳ tác phẩm mới nào cho tôi. Drain có nghĩa là không đưa cho tôi bất kỳ công việc mới nào và loại bỏ công việc mà tôi đang có. Vì vậy, đó là một trong những chúng tôi muốn. Lưu ý rằng chúng tôi cũng có thể cập nhật vai trò ở đây nếu chúng tôi muốn. Vì vậy, việc thăng hạng chỉ là đường cú pháp trên đầu lệnh cập nhật này. Được rồi, hãy đóng tab này, quay lại đây và chúng ta có thể tạo một node **docker**, sau đó, chúng ta có thể cập nhật, đặt tính khả dụng của mình và đánh dấu nó là đã cạn và đó là lý do tại sao tôi cũng thích hoàn thành tab, nhân tiện , xuống bên dưới. Vì vậy, hãy tiếp tục và chạy điều này, và chỉ cần xem những gì xảy ra ở trên cho cả các node và nhiệm vụ. Và đặc biệt, chúng ta cần trau dồi về m2 trước khi tôi quên điều đó, vì vậy chúng ta cần đặt điều đó lên cuối cùng. Theo dõi dòng cho m2 ngay tại đây, và sau đó, theo dõi và xem điều gì sẽ xảy ra với dịch vụ weby của chúng tôi, hiện đang chạy trên m2. Được rồi, node của chúng tôi đã cạn kiệt và sau đó bạn có thể thấy chúng tôi có các nhiệm vụ đang thay đổi ở phía dưới. Và sau đó, ở trên, chúng ta phải thay đổi thành node m2 của chúng ta. Vì vậy, bạn có thể thấy chúng tôi hiện đang có sẵn cống. Ở bên dưới, bạn có thể thấy rằng nhiệm vụ của chúng ta đã được chuyển từ m2 sang w4 và bạn có thể thấy có một bộ đếm nhỏ đang chạy ở đây. Đây chỉ là một tác dụng phụ của thực tế là chúng ta đang xem xét sự khác biệt và xem lệnh này, lệnh này sẽ in ra thời lượng kể từ khi có điều gì đó xảy ra nếu nó là gần đây. Quan trọng nhất, chúng tôi hiện đang chạy trên w4, w4 tác vụ weby của chúng tôi. Vì vậy, hãy xác nhận điều đó trong trình duyệt của chúng tôi. Của bạn đây. Chúng tôi đã chuyển nhiệm vụ weby của mình sang đây sang node w4. Tôi nên nói vùng nhiệm vụ vì bản thân nhiệm vụ đã kết thúc vòng đời của nó trên m2 khi chúng ta thiết lập mọi thứ để tiêu hao và một nhiệm vụ mới đã được tạo ở đây trên w4. Vì vậy, các nhiệm vụ là một thỏa thuận một lần . Chúng không thực sự di chuyển xung quanh. Tuy nhiên, khe, là ý tưởng mà tôi có, x bản sao, vì vậy tôi có x lô. Và trong trường hợp này, tôi không quan tâm họ đang ở đâu, nhưng họ phải ở đâu đó. Đó là những gì di chuyển xung quanh. Và cuối cùng, chúng tôi nhận được một volume mới ở đây. Và, tất nhiên, điều tuyệt vời là nếu tôi truy cập trình duyệt và làm mới trang web cho ứng dụng đó là một phần của dịch vụ mà chúng tôi vừa giới thiệu, đối với người dùng cuối, không có gì để cập nhật. Tôi không phải thay đổi node nào mà tôi truy cập để truy cập dịch vụ cụ thể đó, ngay cả khi nó được di chuyển xung quanh trên cụm của tôi. Đó là một điều rất tiện lợi. Và đó là một tính năng của cái được gọi là bộ cân bằng tải bầy đàn xâm nhập. Vì vậy, khi lưu lượng truy cập vào cụm của bạn, bạn có thể truy cập vào bất kỳ node nào. Nó sẽ được cân bằng tải ở bất kỳ nơi nào dịch vụ đang chạy. Vì vậy, khi tôi nói rằng có một quy trình điều chỉnh liên tục chạy để đảm bảo rằng trạng thái mong muốn được thực thi, đó là ý tôi bằng cách thực hiện thay đổi đối với node, m2, điều này đã kích hoạt thay đổi đối với nhiệm vụ dịch vụ của chúng tôi. Bây giờ nếu chúng ta sắp xếp và triển khai các container theo cách thủ công, chúng ta sẽ phải xoay container và xoay nó ở một nơi khác, và sau đó chúng ta phải làm điều đó cho mọi container chạy trên m2. Và nếu việc di chuyển những volume đó gây ra bất kỳ vấn đề nào như tải quá nhiều trên một máy nhất định, chúng ta có thể phải di chuyển các volume khác xung quanh để lắp mọi thứ vào nếu chúng tôi có một volume nhiều công việc trên m2. Và, tất nhiên, vào thời điểm đó, chúng tôi cũng sẽ không có người quản lý vì chúng tôi sẽ là người quản lý, và đó là lý do tại sao điều này vô cùng mạnh mẽ. Nó giống như **Docker** Compose của bạn, cụ thể là tệp ngăn xếp của bạn, chỉ cần tưởng tượng nó được tải lên bộ nhớ và nó liên tục được kéo để đảm bảo rằng trạng thái trong tệp ngăn xếp của bạn khớp với trạng thái trên cụm thực của bạn.

**Swarm**Kit có một trong những cơ sở mã được tài liệu hóa tốt nhất từng có

Tôi muốn chia sẻ một nguồn thực sự hữu ích. Vì vậy, có rất nhiều tài liệu cho **Docker** bên trong tài liệu **Docker**. Chúng tôi đã thấy một số điều này. Tuy nhiên, đôi khi tôi thấy rằng những lời giải thích không đủ cho những gì tôi đang tìm kiếm hoặc tôi thấy chúng mơ hồ và tôi muốn giải thích thêm một chút. Một trong những tài nguyên tốt nhất mà tôi tìm thấy là bản thân cơ sở mã **Swarm**Kit. Tôi nghĩ đây là một trong những tài liệu hay nhất về tài liệu nội bộ, cơ sở mã mà tôi từng thấy. Tôi muốn cho bạn thấy điều đó một cách ngắn gọn. Vì vậy, nếu bạn đến đây và chỉ xem qua các tệp trên GitHub một mình, điều đó có lẽ là đủ tốt. Bạn cũng có thể sao chép nó vào máy của mình, nhưng bạn đã có mọi thứ ở đây và GitHub khá dễ sử dụng trong những ngày này. Vì vậy, nếu bạn nhấn T trên bàn phím, bạn có thể tìm kiếm tệp và sau đó .proto là một trong những đề xuất số một mà tôi tìm kiếm trợ giúp về mặt giải thích vì đây là thứ được sử dụng để mô tả rất nhiều giao diện mà bạn và tôi tương tác vào cuối ngày, tôi nên nói công cụ **Docker** tương tác, nhưng chúng tôi tương tác với công cụ **Docker**, vì vậy các API được xác định ở đây. đặc biệt, hãy xem bên trong thư mục API. Ngoài ra còn có các định nghĩa cho API giữa các node worker và manager, cách xử lý công việc. Và vì vậy nếu bạn đi qua đây khi bạn cần tìm hiểu về mọi thứ, đôi khi có một lời giải thích tốt hơn. Một vài điểm mà tôi thấy thực sự hữu ích, số một trong số đó là điểm thông số kỹ thuật. Tôi đã mở nó ra trong một tab ở đây. Vì vậy, bạn có thể thấy đây thực sự là một phần của API mà chúng tôi tương tác bởi vì nó được định nghĩa là các đối tượng vùng chứa cho đầu vào do người dùng cung cấp , do đó sẽ là từ công cụ **Docker**. Vì vậy, giả sử tôi tò mò về chính xác thoát nước nghĩa là gì. Tôi chỉ có thể nhập vào đây thoát và bởi vì đây là một API mà chúng tôi tương tác, nó có thể sẽ có định nghĩa mà tôi muốn. Vì vậy, chúng tôi đã có bảng liệt kê Tính khả dụng của chúng tôi ở đây trong thông số kỹ thuật của node này. Vì vậy, nó trông giống hệt như những gì chúng tôi đang tìm kiếm. Chúng tôi đã có Hoạt động. Chúng tôi đã tạm dừng. Chúng tôi đã có Drain. Vì vậy, đó là những tùy chọn chính xác mà chúng ta đã thấy trong dòng lệnh. Và ở trên đây, chúng tôi có những lời giải thích nhỏ hay ho này. Các node thoát bị tạm dừng và mọi tác vụ đang chạy trên chúng sẽ bị loại bỏ. Vì vậy, điều đó đã mở ra cánh cửa để có thể tìm kiếm một số thuật ngữ khác như, đuổi khỏi có nghĩa là gì? Hay nhiệm vụ nghĩa là gì? Hoặc chúng tôi có thể tìm kiếm nhiệm vụ bên trong đây nếu chúng tôi muốn. Nhiệm vụ xác định mẫu tác vụ. Dịch vụ này sẽ sinh ra. Nó khá tuyệt. Vì vậy, đây là trên thông số dịch vụ. Đây là định nghĩa về một dịch vụ. Vì vậy, một điều tôi sẽ nói với bạn, một phần thông tin quan trọng, khi bạn đang chạy các lệnh đặc tả cho chức năng **Swarm**Kit, như với weby\_we bằng cách kiểm tra dịch vụ ở đây, bất cứ thứ gì bạn thấy bên trong đây là những thứ bạn có thể tìm kiếm bên trong tệp .proto đó hoặc một nơi nào khác bên trong cơ sở mã đó. Vì vậy, nếu tôi tò mò về UpdateStatus, chúng ta có thể xem xét liệu đó có phải là bên trong đây không. Có thể không phải lần nào cũng thành công, nhưng tôi cũng có thể xem lại dòng lệnh. Tôi có một bản sao của **Swarm**Kit được kéo xuống đây. Tôi có thể chạy qua và tìm kiếm UpdateStatus và xem những gì tôi nghĩ ra. Vì vậy, tôi nhận được rất nhiều lượt truy cập ở đây. Tôi thậm chí đã xem một số tài liệu với một số trong số những lần truy cập này. Chúng ta bắt đầu. UpdateStatus chứa trạng thái của bản cập nhật, nếu có. Và có vẻ như tôi đã cắt bớt một dòng ở đây. Đây là trong tệp objects.proto, vì vậy tôi có thể mở nó lên. Tôi quan tâm đến dòng 127, vì vậy chúng ta bắt đầu. Được rồi, nếu một cái đang được xử lý. Vì vậy, tệp objects.proto này chứa các định nghĩa về những thứ nằm trên thông số dịch vụ mà chúng tôi nhận được từ quá trình kiểm tra dịch vụ **Docker** của chúng tôi. Có UpdateStatus của chúng tôi. Đây là điểm cuối là gì. Một khía cạnh hữu ích khác đối với tài liệu này là chính mã.

Mã của **Swarm**Kit có thể truy cập được - Hãy kiểm tra!

Tôi thấy rằng nó không chỉ là tài liệu ở đây. Tôi thực sự nghĩ rằng bản thân mã này rất dễ đọc. Ví dụ, đây là tệp worker.go, vậy đây là tệp worker. Vì vậy, việc triển khai công nhân cụ thể này đảm nhận việc quản lý nhiệm vụ cốt lõi, tất nhiên, là toàn bộ công việc của người lao động để nhận nhiệm vụ và thực hiện chúng. Vì vậy, nếu bạn muốn tìm hiểu cách hoạt động, đây có thể là một trong những tệp cần xem xét. Vì vậy, đây là một phần của nó, đọc các tác vụ từ cơ sở dữ liệu và bắt đầu bất kỳ trình quản lý tác vụ nào có thể cần thiết. Vì vậy, có một số khái niệm về một người quản lý nhiệm vụ. Nếu tôi muốn tìm hiểu sâu hơn một chút, tôi có thể tìm kiếm điều đó và bắt đầu hiểu việc triển khai các tính năng khác nhau của **Swarm**Kit. Tuy nhiên, một trong những mục yêu thích của tôi và tôi đã mở điều này ra đây trên trang web, đây là người điều phối và đó là một API được cung cấp bởi một nhóm người quản lý để các đại lý kết nối. Vì vậy, các đại lý kết nối với người điều phối này, họ nhận các nhiệm vụ được giao theo cách này, và sau đó họ báo cáo trạng thái theo cách này. Các loại là một yêu thích khác của tôi. Ví dụ: nếu bạn có một nhiệm vụ ở một trạng thái nhất định và bạn không biết điều đó có nghĩa là gì như đang chờ xử lý, chẳng hạn, đây là định nghĩa của việc chờ xử lý, chờ quyết định lên lịch. Vì vậy, vào thời điểm đó, chúng tôi biết rằng công việc của chúng tôi thậm chí vẫn chưa được lên kế hoạch. Và vì vậy, chúng ta phải hỏi, à, điều gì có thể khiến nhiệm vụ của chúng ta đang chờ xử lý, không được lên lịch trình? Một trường hợp có thể là không có sẵn tài nguyên vì có một số ràng buộc không cho phép đặt nó ở bất kỳ đâu trên bộ máy hiện tại mà chúng tôi có, vì vậy chúng tôi đã hạn chế nó quá mức. Và nơi cuối cùng của tôi để đến, một trong những mục yêu thích tuyệt đối của tôi, là tệp lập lịch, và đây là logic để thực sự lên lịch cho công việc. Nếu bạn thực sự muốn hiểu khía cạnh hòa giải, cách quản lý xảy ra với bạn, điều phối, tất cả các nhiệm vụ bạn sẽ phải làm nếu bạn muốn tự đặt máy hoặc đặt các volume theo cách thủ công, tất cả logic đó đều có ở đây để hiểu và Tôi nghĩ điều đó thật dễ hiểu vì tôi nghĩ bạn có thể hiểu khái niệm SSHing vào máy móc và khởi động vùng chứa ở nhiều vị trí khác nhau nếu bạn cố gắng ghép một cụm theo cách thủ công. Vì vậy, hãy sử dụng nó như một nguồn tài nguyên. Khi bạn có thời gian và khi bạn sử dụng **Swarm**Kit ngày càng nhiều, đừng ngại tìm hiểu chính cơ sở mã và ít nhất hãy bắt đầu với các định nghĩa và các nhận xét nhỏ ở đây và ở đó. Ở đây, ví dụ, là vòng lặp sự kiện của bộ lập lịch, vì vậy đây là một phần chức năng cốt lõi. Điều này chỉ lặp đi lặp lại, thực hiện cùng một công việc để đảm bảo rằng cụm đã hội tụ đến trạng thái mong muốn của nó.

Tài liệu thiết kế Repo của **Swarm**Kit được lấp đầy với Kiến trúc và Điều khoản

Một phần hữu ích khác trong kho **Swarm**kit là thư mục thiết kế. Bên trong đây, bạn sẽ tìm thấy một loạt các tệp đánh dấu đề cập đến thuật ngữ và trong các trường hợp khác là giải thích về chức năng. Ví dụ: bạn có thể tìm hiểu về bè và cách nó được sử dụng và triển khai trong hai tệp này. Nó được sử dụng bởi các node của trình quản lý như một giao thức để giao tiếp trạng thái cụm để sao chép nó và đảm bảo rằng nó luôn được bảo toàn hoặc có thể là mức cao hơn một chút nếu bạn tò mò về mô hình tác vụ, cách hoạt động chính xác. Chà, đó là một lời giải thích chi tiết thực sự hay ở đây, không dài lắm, nó giải thích từng phần trạng thái trong một nhiệm vụ, mục đích của nó là gì, thậm chí đi vào vòng đời tác vụ, lịch sử tác vụ, mà chúng ta đã nói một chút về , và sau đó cách xuống bên dưới là mô hình vị trí và đó là những gì tôi đã nói đến khi tôi đề cập đến vị trí nhiệm vụ, đó là .1.2, v.v. phụ thuộc vào số lượng bản sao bạn có cho một nhiệm vụ. Ngoài ra còn có một tài liệu danh pháp tốt đẹp. Tôi nhận thấy rằng các thuật ngữ bên trong đây là ngắn gọn và thường sâu sắc trong việc chỉ ra các thuật ngữ khác sẽ giúp hiểu chúng. Ví dụ: chỉ cần nhìn vào agent ở đây, chúng ta có thể thấy agent điều phối việc điều phối, vì vậy chúng ta đã thấy điều phối viên trước đó trong mã công việc của một nhân viên. Vì vậy, tác nhân duy trì một kết nối với người điều phối và sau đó không chỉ các nhiệm vụ đến từ người điều phối, mà người đại diện quay lại và nói với người điều phối chính xác điều gì đang xảy ra với một nhiệm vụ nhất định để nó cung cấp trạng thái trở lại cho người điều phối để có thể được tổng hợp ở cấp độ cụm và đó là cách chúng tôi có quyền truy cập vào thông tin đó khi chúng tôi chạy các truy vấn dòng lệnh của mình. Hoặc chúng ta có thể thấy, ví dụ, một tác vụ đang chạy, khi nào nó bắt đầu chạy và điều đó cung cấp cho bạn tính toán xem nó đã chạy trong bao lâu hoặc nếu có lỗi, hỏng hóc, điều gì đã xảy ra.

Tạo tệp ngăn xếp để triển khai dịch vụ viz của chúng tôi

Được rồi, đối với ví dụ tiếp theo này, tôi muốn chuyển sự chú ý của chúng ta sang một tình huống khác. Tôi muốn nhìn lại dịch vụ viz của chúng tôi . Vì vậy, trước đây, chúng tôi đã tạo điều này bằng lệnh tạo dịch vụ **docker**, bây giờ tôi muốn xem nó trông như thế nào dưới dạng một ngăn xếp. Nếu bạn muốn một thử thách, bạn có thể tạm dừng video và cố gắng tự mình chuyển đổi video này thành một ngăn xếp và xem bạn nghĩ ra gì, sau đó tiếp tục với phần còn lại của chúng tôi. Chúng ta hãy đi qua điều này. Tôi không thực sự coi tệp ngăn xếp là phần quan trọng. Đó là một bài tập tốt, nhưng nó là một tệp ngăn xếp khá tầm thường. Tôi đã tạo nó ở đây rồi. Chúng ta hãy đi qua điều này một cách nhanh chóng. Chúng tôi có image của chúng tôi. Chúng tôi có các cảng của chúng tôi. Chúng tôi có nhiều tập. Bây giờ, điều này đơn giản hơn một chút, tôi nghĩ, hơn là ở dòng lệnh với dịch vụ **docker** tạo nơi chúng ta cần sử dụng cú pháp mount dài hơn. Trong tệp ngăn xếp, chúng tôi chỉ có thể sử dụng tốc ký truyền thống, nếu bạn muốn, cú pháp khối lượng bởi vì chúng tôi chỉ đang gắn vào ổ cắm của **docker**. Điều đó nói rằng, đó là một điểm cộng cho cách tiếp cận tệp ngăn xếp vì tôi nghĩ rằng cách này dễ đọc hơn. Sau đó, chúng ta hãy chia màn hình ở đây. Chúng tôi đặt dịch vụ của mình lên hàng đầu vì tôi muốn nói về ràng buộc của vị trí. Đó thực sự là điều mới lạ ở đây liên quan đến các ngăn xếp. Đây là một tính năng mà bạn sẽ không tìm thấy bên trong **Docker** Compose và đó là phần triển khai hoặc phần tử bên trong tệp cấu hình được lồng bên dưới dịch vụ của bạn. Trong trường hợp này, tôi gọi nó là web. Vì vậy, bên dưới triển khai, chúng tôi có vị trí. Tôi đánh giá cao cách đọc này. Nó triển khai với một ràng buộc về vị trí để đảm bảo rằng vai trò của node là người quản lý. Tôi nghĩ rằng nó đọc thực sự độc đáo khi chỉ lướt qua mọi thứ ở đó, ngay cả đối với một người mới sử dụng tệp ngăn xếp, trái ngược với cách tiếp cận dòng lệnh nơi chúng ta chỉ truyền một đối số ràng buộc và bạn có thể tự hỏi, chính xác thì chúng ta là gì hạn chế? Và tất nhiên, có khá nhiều khía cạnh khác có thể được lồng vào bên dưới phần tử triển khai này và đó là nơi bạn sẽ tìm thấy hầu hết những gì mới lạ về tệp ngăn xếp về cấu hình của dịch vụ. Vì vậy, không có gì quá lạ mắt ở đây bởi vì chúng ta đã trải qua kịch bản xử lý một hạn chế bị thiếu. Hãy nhớ rằng, chúng tôi không có trình quản lý vai trò node để bắt đầu và điều đó gây ra một số vấn đề khi nhiệm vụ của chúng tôi kết thúc trên một node công nhân mà nó rõ ràng không có quyền truy cập vào tập dữ liệu của trình quản lý bầy đàn. Vì vậy, chúng tôi đã xử lý hạn chế bị thiếu đó. Bây giờ chúng ta có một ràng buộc khác, và cái này tôi nghĩ là thực sự thú vị vì nó là một ràng buộc ngầm, những ràng buộc không quá rõ ràng so với cấu hình thực tế. Nó không phải là có bất cứ điều gì bị thiếu trong cấu hình. Tôi muốn bạn thấy sức mạnh của nó như thế nào chỉ cần thay đổi một chút tệp cấu hình này và cách **Docker** **Swarm** phản ứng với điều đó và thích ứng với môi trường động để triển khai thành công ứng dụng sau lần đầu tiên không thành công.

Khởi tạo một bầy Pi trong vòng chưa đầy 3 phút bằng cách sử dụng thiết lập kim loại trần của 5 Raspberry PI!

Vì vậy, chúng ta hãy nói một chút về môi trường trước. Nếu tôi kéo danh sách ngữ cảnh của mình lên, bạn sẽ có thể thấy ngay bây giờ tôi đang kết nối với pi1 và tôi có từ pi1 đến pi5, và đó là vì tôi đã thiết lập một cụm Pi để chúng tôi sử dụng và tất cả các node đã cài đặt **Docker**. Tôi muốn chúng tôi tiếp đất chạy, nhưng tôi đã bỏ qua một điều, tôi không khởi tạo bầy đàn vì tôi nghĩ rằng đây sẽ là cơ hội tốt để thực hành một cái gì đó mà chúng tôi đã thử ở đầu khóa học, thực sự chỉ để xem dễ dàng và mạnh mẽ như thế nào để thiết lập nó trong một môi trường hoàn toàn mới . Vì vậy, tôi có thể cho bạn thấy rằng môi trường không được thiết lập. Tôi có một lệnh khá dài ở đây. Bạn phải tin tưởng tôi về điều này. Về cơ bản, điều này lặp lại bằng cách sử dụng cú pháp Z shell năm số dẫn đến việc tôi gọi **docker** và thông tin về mỗi node Pi khác nhau của chúng tôi. Vì vậy, chúng tôi đã có pi1, pi2, pi3, pi4, pi5 bằng cách sử dụng lập luận ngữ cảnh ở đây và tôi chỉ yêu cầu lấy thông tin ra bằng cách định dạng với JSON để chỉ lấy thông tin bầy đàn và định dạng nó độc đáo với jq. Vì vậy, chúng ta hãy xem xét điều đó. Bạn có thể thấy tất cả các node ở đây trong một giây sẽ chuyển sang trạng thái không hoạt động. Được rồi, tất cả năm trong số đó đều được đánh dấu là không hoạt động, vì vậy chúng tôi có một điểm khởi đầu tốt để khởi tạo bầy đàn. Vì vậy, tôi đã kết nối với pi1. Tại sao chúng ta không sử dụng nó làm node đầu tiên? Bạn có nhớ tôi gõ gì ở đây để tạo một bầy trên node đầu tiên không? Vì vậy, đó là lệnh **docker** **Swarm** init, và sau đó bạn có nhớ những gì chúng ta cần chuyển cho điều này? Vì vậy, chúng ta cần thông qua địa chỉ được quảng cáo, địa chỉ mà những người quản lý sẽ sử dụng để nói chuyện với nhau. Nếu tôi muốn nhận được điều đó, tôi có thể sử dụng một cuộc gọi nhanh tới SSH. Tôi cũng đã thiết lập SSH cho mỗi node này. Vì vậy, đó là pi1.lan và sau đó tôi có thể chạy một lệnh như lấy địa chỉ IPv4, hiển thị địa chỉ và tôi có thể xem qua đây và tôi biết rằng đây là địa chỉ mà tôi muốn ở đây. Đây là 2.101. Vì vậy, bây giờ tôi có thể quay lại và tôi có thể chạy init **Swarm** **docker** của mình và tôi có thể chuyển địa chỉ được quảng cáo của mình và dán vào 2.101 đó. Khi tôi chạy, tôi đã kết nối với ngữ cảnh của p1. Bam, chúng tôi có một cụm được tạo giống như vậy. Tôi có thể làm một node **docker** ls và sẽ có một node ở đây. Tôi luôn ấn tượng với tốc độ tạo ra một môi trường an toàn vì các node này thực sự đang sử dụng thông tin liên lạc được mã hóa ngay từ điểm xuất phát và đó là những gì các mã thông báo đó phải làm. Chúng là các mã thông báo mật mã giúp bảo mật thông tin liên lạc, cũng như bảo mật các node mới khi chúng tham gia. Vì vậy, bạn cũng sẽ muốn bảo vệ những mã thông báo đó. Bây giờ chúng ta có một số đầu ra ở đây cung cấp cho chúng ta lệnh chạy để kết nối một worker. Tôi chỉ định làm rõ điều này bởi vì tôi muốn bắt đầu bằng cách kết nối hai và ba với tư cách là người quản lý. Và để làm điều đó, tôi sẽ cần mã thông báo tham gia. Hãy giả sử tôi quên viết nó ra hoặc không truy vấn nó ngay từ điểm bắt đầu khởi tạo node đầu tiên. Bạn có nhớ tôi làm cách nào để đạt được điều đó không? Vì vậy, tôi cần được kết nối với bối cảnh cho một trong những người quản lý hiện tại. Tôi có thể thực hiện một nhóm **docker** và sau đó tham gia - mã thông báo, và sau đó tôi chỉ cần chỉ định người quản lý hoặc nhân viên, trong trường hợp này tôi muốn mã thông báo người quản lý. Tôi đoán trong khi làm việc đó, tôi cũng có thể lấy được mã thông báo của công nhân. Vì vậy, đó là cách bạn có thể truy cập vào những thứ đó nếu bạn quên chúng là gì. Vì vậy, chúng ta hãy tiếp tục sao chép và dán mã thông báo người quản lý, toàn bộ dòng ở đây, đó là toàn bộ lệnh gọi đến mã thông báo tham gia ‑‑ **docker** **Swarm** và chúng ta chỉ cần dán nó vào bảng điều khiển ở đây. Và tôi sẽ quay lại phần bắt đầu ở đây và tôi sẽ thay đổi ngữ cảnh thành pi2 bởi vì tôi đã thiết lập tất cả bối cảnh để trỏ đến các node khác nhau này. Vì vậy, bằng cách làm điều đó, tôi sẽ yêu cầu pi2 tiếp tục và tham gia với tư cách là người quản lý, điều hành nó, và bạn sẽ làm được. Và sau đó tôi có thể quay lại đây và tôi có thể chạy nó cho pi3, và bạn cứ việc. Chúng tôi đã thêm ba người quản lý vào nhóm bầy đàn của chúng tôi. Nếu chúng ta tò mò muốn xem thì sẽ có node **Docker** và chúng ta đã có tất cả ba node. Và bây giờ tôi muốn tạo hai cái cuối cùng dưới dạng công nhân, vì vậy tôi sẽ lấy câu lệnh tham gia mã thông báo công nhân ‑‑ cho điều đó, dán nó vào, chuyển sang phần bắt đầu, thay đổi ngữ cảnh trên này thành pi4, và sau đó chúng ta ' Tôi cũng sẽ thay đổi điều này để chạy cho pi5. Cứ như vậy, chúng tôi đã thiết lập toàn bộ cụm của mình. Điều đó không phải là khá ấn tượng? Và một lý do khác mà tôi muốn thể hiện điều này là vì khi chúng tôi mới bắt đầu, tôi muốn giải thích một số cơ sở khái niệm nên không chỉ tôi ném điều này vào bạn. Bây giờ tôi đang ném vào bạn như một sự bồi dưỡng, cũng như một cơ hội để xem việc thiết lập một cụm có thể dễ dàng như thế nào. Được rồi, hãy lấy tệp ngăn xếp đó và triển khai nó vào cụm của chúng ta.

Khắc phục sự cố Dịch vụ Không thành công ở Cấp Tác vụ với `**Docker** Stack ps --no-trunc`

Được rồi, vậy là các node của chúng ta đã sẵn sàng và tôi đang ngồi bên trong thư mục ví dụ với các tệp viz, đặc biệt là ngăn xếp viz ở cuối. Bối cảnh cũng được thiết lập để sử dụng pi1, vì vậy tôi đã sẵn sàng triển khai ngăn xếp image hóa của mình cho cụm Pi mới tạo của tôi. Bạn có nhớ cách chúng tôi triển khai một ngăn xếp không? Chỉ cần nhập triển khai ngăn xếp **docker** và sau đó chúng ta cần chỉ định tệp cho ngăn xếp của mình, vì vậy viz - ngăn xếp trong trường hợp này. Và sau đó chúng ta cũng cần đặt tên cho ngăn xếp, vì vậy chúng ta cũng sẽ gọi nó là viz. Được rồi, điều đó đã được triển khai và bạn có thể thấy rằng dịch vụ và mạng của chúng tôi đã được tạo. Bạn có nhớ cách chúng tôi nhìn vào ngăn xếp của chúng tôi không? Từ dòng lệnh, chúng ta đã có lệnh ngăn xếp **docker**, chúng ta có thể liệt kê các ngăn xếp của mình. Và bạn nhé, chúng tôi đã có một ngăn xếp viz của chúng tôi, có vẻ như nó đang hoạt động, ít nhất là nó đã được người quản lý công nhận. Làm thế nào để chúng ta tìm hiểu và xem điều gì đang thực sự xảy ra? Chà, liên quan đến các dịch vụ của chúng tôi, chúng tôi xem xét những dịch vụ đó ở cấp độ thấp hơn một chút. Vì vậy, chúng tôi có thể đi xuống cấp độ dịch vụ, xem xét những thứ đó và bạn có thể thấy chúng tôi có viz\_web của mình. Và vì vậy \_web thực tế là tệp ngăn xếp, tôi sử dụng web tên cho dịch vụ, vì vậy nó sẽ đặt nó vào cuối tên ngăn xếp mà tôi đã chọn. Vì vậy, có dịch vụ của chúng tôi, và ít nhất theo như liên quan đến màn hình này, nó không có vẻ gì là sai ngoại trừ việc chúng tôi có 0 trên 1 bản sao, và đó là một vấn đề. Vì vậy, chúng tôi đang chạy cái được gọi là loại dịch vụ sao chép. Nó có thể mở rộng đến bất kỳ số nào mà chúng tôi chỉ định mà cụm của chúng tôi có thể xử lý. Và hiện tại, chúng tôi chỉ có một phiên bản đang chạy và nó chưa hoạt động. Bây giờ có lẽ đó chỉ là sự chậm trễ trong quá trình lấy image và bắt đầu mọi thứ, vì vậy chúng tôi luôn có thể lấy lại trạng thái dịch vụ của mình để xem điều gì đang xảy ra. Bạn có thể thấy rằng chúng tôi vẫn chưa có một phiên bản dịch vụ nào của chúng tôi được thiết lập và chạy, vì vậy đó sẽ là một nhiệm vụ và cụ thể là một vùng chứa. Vì vậy, một cái gì đó có lẽ là không đúng ở đây. Làm thế nào chúng ta có thể xem xét cấp độ nhiệm vụ cho ngăn xếp của chúng ta? Có hai cách. Chúng ta có thể chuyển sang dịch vụ **docker** và sau đó là ps, nhưng đừng quên rằng chúng ta cũng có ngăn xếp **docker** và ps. Cái này chỉ yêu cầu tên của ngăn xếp và nó sẽ in ra tất cả các tác vụ cho ngăn xếp đó trên tất cả các dịch vụ. Nếu chúng tôi đi xuống cấp dịch vụ, chúng tôi sẽ chỉ nhận được các nhiệm vụ cho dịch vụ. Bây giờ chúng ta có thể muốn điều đó nếu chúng ta chỉ gặp vấn đề với một dịch vụ trong số nhiều dịch vụ trong một ngăn xếp nhất định. Được rồi, vì vậy chúng tôi có thêm một số thông tin ở đây. Đây là nơi tôi sẽ cung cấp cho bạn một mẹo nhỏ về lệnh ps ngăn xếp **docker**, và nhiều lệnh ps hoặc danh sách trong **Docker**, có tùy chọn no - trunc, viết tắt của không cắt bớt. Nếu chúng tôi chạy điều đó, chúng tôi sẽ nhận được thêm thông tin chi tiết về những gì sai, cũng như chi tiết dài hơn về thông báo của chúng tôi trên image của chúng tôi, mà chúng tôi có thể không thực sự quan tâm, cũng như ID đầy đủ, mà chúng tôi cũng không ' Tôi không quan tâm đến. Tuy nhiên, điều chúng tôi quan tâm là một số chi tiết về những gì đang xảy ra, bạn có thể xem ở đây, chúng tôi không có node phù hợp, nền tảng không được hỗ trợ trên 3 node, các ràng buộc về lịch trình không được thỏa mãn trên 2 node. Bạn nghĩ phần này có nghĩa là gì, các ràng buộc lập lịch không thỏa mãn trên 2 node? Vâng, hãy nhớ rằng, với tệp ngăn xếp của chúng tôi, chúng tôi có một hạn chế về vai trò là người quản lý, và do đó hai trong số các node là công nhân và do đó, không đủ là node mà điều này có thể chạy trên đó. Sau đó, liên quan đến ba người còn lại, bạn nghĩ điều gì có thể sai với những điều đó? Vâng, nó nói rằng nền tảng không được hỗ trợ, và đó là gợi ý cụ thể về điều gì đó. Image mà chúng tôi đang cố gắng chạy trên các node khác nhau là trình quản lý này, nó được biên dịch cho một kiến trúc CPU khác với các node Pi. Các node Pi có bộ xử lý ARM. Nếu chúng tôi muốn xác nhận rằng chúng tôi có thể lấy thông tin hệ thống **docker**, hãy đổ thông tin này ra đây, nó sẽ cung cấp cho chúng tôi phía máy khách cũng như phía máy chủ, nhưng chúng tôi chỉ quan tâm đến phía máy chủ ở đây. Vì vậy, tôi có khách hàng của tôi ở trên, chỉ cần bỏ qua điều đó. Dưới đây ở phía máy chủ, sẽ có OSType và Kiến trúc, vì vậy bạn có thể thấy chúng tôi cần armv7L, về cơ bản armv7 là những gì chúng tôi đang tìm kiếm liên quan đến image **Docker**.

Tìm các Nền tảng được Hỗ trợ của Image trên **Docker** Hub hoặc qua `thẻ hub-tool ls --platforms`

Chính xác những gì image có sẵn? Chà, một vài cách để có được thông tin đó. Vì vậy, nếu chúng ta sử dụng trình duyệt, có lẽ cách dễ nhất là đến với **Docker** Hub và bạn có thể khám phá image hoặc bạn có thể nhập / r / và sau đó đi ra ngoài và sao chép slug cho image, dán nó vào trình duyệt và điều đó sẽ kéo lên kho lưu trữ. Đây là bản dựng chính thức của ví dụ này theo như các ví dụ về **docker**s, tổ chức trên **Docker** Hub có liên quan. Bên dưới Thẻ sau đó, chúng ta có thể đến và xem thẻ mới nhất. Và nếu chúng ta nhìn vào điều này, chúng ta có thể thấy rằng kiến trúc hệ điều hành được đặt thành AMD64, vì vậy điều đó sẽ không hoạt động tốt với bộ xử lý ARM. Nếu bạn quan tâm đến cách tiếp cận dòng lệnh để có được thông tin nền tảng tương tự, một trong những cách yêu thích của tôi để làm điều đó là với trung tâm thử nghiệm mới - công cụ CLI. Nếu chúng tôi nhận được trợ giúp cho lệnh này, chúng tôi có thể thấy có một loạt các lệnh con và nhân tiện, bạn sẽ cần xác thực để sử dụng lệnh thẻ mà chúng ta sẽ sử dụng trong giây lát. Một trong những lệnh con của lệnh này và nhân tiện, tôi tin rằng công cụ Hub CLI này cuối cùng sẽ tìm thấy đường vào **Docker** CLI. Nó chỉ là thử nghiệm tại thời điểm này và nó đi kèm với **Docker** Desktop theo mặc định, vì vậy bạn có thể sử dụng nó nếu bạn có **Docker** Desktop. Tuy nhiên, có một lệnh con được gọi là thẻ, mà chúng ta muốn xem xét và bên dưới đó là một loạt các lệnh con , số một trong số đó là ls. Điều này sẽ giúp chúng tôi liệt kê ra các thẻ cho một kho lưu trữ nhất định, vì vậy những gì chúng tôi có thể làm là cung cấp cùng một **docker**samples / visualizer đó. Và ở đó bạn đi. Chúng tôi có hai thẻ riêng biệt được liệt kê, nhưng chúng tôi không có bất cứ điều gì về kiến trúc CPU. Vì vậy, những gì chúng ta có thể làm, tôi sẽ chia ra một tab mới ở đây, chúng ta có thể lấy trợ giúp cho lệnh này và bên dưới đây bạn có thể thấy có một số tùy chọn, một trong số đó là bao gồm các nền tảng. Vì vậy, tôi có thể giải quyết vấn đề đó vào các tùy chọn cho lệnh ls và khi tôi làm điều đó, bạn có thể thấy ở phía bên tay phải , chúng tôi cũng có kiến trúc được liệt kê là AMD64.

Thay đổi image thành weshigbee / **Swarm**gs2-viz: Chỉ cần chỉnh sửa tệp ngăn xếp và triển khai!

Vì vậy, những gì tôi đã làm để làm cho điều này thành công, tôi đã tiếp tục và tạo ra image của riêng mình có hỗ trợ cho nhiều nền tảng. Có lẽ tôi sẽ cố gắng gửi cái này sau khóa học của mình, vì tôi vừa chia nhỏ kho lưu trữ trực quan hóa bầy đàn tồn tại và nếu không, tôi chắc chắn ai đó sẽ làm được. Vì vậy, nếu bạn thực sự thấy rằng trình hiển thị các ví dụ về bác sĩ có hỗ trợ cho nhiều nền tảng trong tương lai, đừng nhầm lẫn, đó chỉ là tại thời điểm ghi lại. Tôi đề cập đến điều đó bởi vì chúng tôi đã chạy tất cả các lệnh đó tại dòng lệnh và nhìn vào trang web, và chúng tôi có thể thấy rằng nó chỉ có hỗ trợ AMD 64 và điều đó có thể thay đổi. Cái này nằm ở đây tại weshigbee / **Swarm**gs2 - viz. Nếu bạn nhìn vào các thẻ ở đây, bạn sẽ thấy ngay tại đây, tôi có các kiến trúc khác nhau được hỗ trợ. Tuy nhiên, tôi thực sự thích khoan ở đây, bởi vì bạn có thể thả xuống để xem trong bản in lớn những kiến trúc nào được hỗ trợ. Và, tất nhiên, từ dòng lệnh, chúng tôi cũng có thể xem thông tin tương tự, chúng tôi có thể dán nó vào kho lưu trữ nơi tôi có những image này và bạn sẽ thấy sự khác biệt ở đây, về mặt kiến trúc hệ điều hành mà chúng tôi có có linux / arm / v7 được liệt kê ngoài linux / amd64, vì vậy điều này có thể chạy cả trên máy Mac của tôi hoặc trên các Raspberry Pi này. Thật thú vị, trên máy Mac của tôi với **Docker** Desktop, tôi có thể chạy bất kỳ nền tảng nào trong số này, nhờ hỗ trợ \_\_\_\_\_ trong **Docker** Desktop. Nếu bạn thực sự tò mò về hỗ trợ đa nền tảng , miễn là kéo image xuống và chạy chúng với một kiến trúc khác, bạn có thể xem bài viết này và tôi cho rằng bạn có thể thiết lập và chạy nó với Raspberry Pi cũng vậy, để bạn có thể chạy image AMD 64 và mô phỏng nó trên Pi's. Vì vậy, hãy ghi nhớ điều đó, nếu bạn đang cố gắng mô phỏng sự cố này trên máy cục bộ của mình và bạn đang sử dụng **Docker** Desktop cho Mac hoặc Windows, thì nó cũng có hỗ trợ đa nền tảng sẵn sàng để bắt đầu và thiết lập, trong khi với phiên bản **Docker** Linux trên Raspberry Pi của tôi thì không, và đó là lý do tại sao tôi có thể gặp sự cố ở đó. Được rồi, quay lại bản demo, bởi vì tôi muốn giữ nguyên giả định rằng tôi chỉ có thể chạy image cánh tay, bởi vì tôi thực sự thích ràng buộc ngầm này xuất hiện và cắn tôi, và ẩn ý là nó không được liệt kê trực tiếp một cách rõ ràng trong chồng tệp. Nó gián tiếp nhờ vào image mà chúng tôi gọi là có hỗ trợ cho một bộ nền tảng nhất định hay không. Vì vậy, những gì chúng ta cần làm sau đó để sửa chữa mọi thứ để ngăn xếp của chúng ta hoạt động? Chà, chúng ta chỉ cần chuyển sang tệp ngăn xếp của mình và thay đổi image. Vì vậy, tôi có thể tiếp tục và dán image của mình ngay trên đầu cho kho lưu trữ của tôi, thẻ này sẽ sử dụng thẻ mới nhất có hỗ trợ đa nền tảng. Bây giờ tôi phải làm gì? Đây là một trong những câu trả lời yêu thích của tôi. Nó sẽ là một triển khai ngăn xếp **docker** khác. Chỉ cần lặp lại cùng một câu lệnh triển khai chính xác mà chúng ta đã có trước đó, giữ nguyên tên ngăn xếp viz và tệp ngăn xếp của chúng ta giống nhau. Tôi nên lưu ý rằng tôi đã lưu tệp ngăn xếp bằng cách này. Nếu bạn gặp bất kỳ sự cố nào với mọi thứ, có thể là bạn đã không lưu tệp ngăn xếp của mình. Được rồi, có vẻ như nó đang cập nhật dịch vụ của chúng tôi viz\_web. Bây giờ đối với cuộn trống, chúng ta hãy xem điều gì xảy ra ở đây, ngăn xếp **docker** ps, nhìn vào ngăn xếp viz của chúng tôi ở đây. Vì vậy, điều này thật thú vị và tôi rất vui vì tôi đã nắm bắt được điều này trước khi mọi thứ thay đổi, vì bạn có thể thấy ngay bây giờ trạng thái mong muốn được đặt thành Đang chạy và trạng thái hiện tại, về cuối cùng, nó được đặt ở đó, được đặt thành Vẫn đang chuẩn bị . Đó là trường hợp thường xảy ra khi bạn kéo một image mất một chút thời gian. Vì vậy, chúng ta hãy xóa và chạy lại nó, **docker** stack ps và viz, và bây giờ bạn có thể thấy trạng thái hiện tại của chúng ta đã hội tụ với trạng thái mong muốn của chúng ta. Nhân tiện, một giờ trước, đó là do tôi quên thêm đoạn ghi âm này vào bản ghi gốc, vì vậy bây giờ tôi đang chỉnh sửa nó và tôi sẽ thêm nó vào. Điều quan trọng là phải thấy rằng trạng thái hiện tại và mong muốn khớp với nhau và tôi thực sự như thực tế là bạn có thể thấy rằng sự hội tụ trong đầu ra ban đầu chưa hoàn toàn hội tụ, mà đôi khi bạn có thể thấy khi bạn kéo một image lần đầu tiên vào cụm của mình hoặc thậm chí kéo nó vào một node trong cụm của bạn cho lần đầu tiên. Hãy nhanh chóng xác nhận những gì chúng ta thấy ở dòng lệnh bằng cách chuyển sang trình duyệt và kéo ứng dụng web trực quan lên. Vì vậy, tôi chỉ cần điều hướng đến 2.101, địa chỉ của Pi đầu tiên, cổng 8080 và có Visualizer của tôi. Tôi cũng có một tên được thiết lập ở đây, pi1.lan, chỉ để truy cập thuận tiện các Raspberry Pi này. Đáng chú ý nhất, mặc dù, tôi muốn chuyển sang một Pi khác và nhìn vào cổng 8080, nhưng trước khi làm điều đó, tôi muốn bạn cho tôi biết những gì sẽ hiển thị ở đây. Vì vậy, tôi đã chỉ vào pi2.lan bây giờ, cổng 8080. Chà, chúng ta sẽ thấy Trình hiển thị. Chúng ta sẽ thấy nó trên mọi node, thậm chí là một trong những node công nhân của chúng ta, vì vậy pi4. Đó là bộ cân bằng tải bầy đàn xâm nhập tại nơi làm việc, giống như thứ mà chúng tôi đã làm trước khi làm việc trên Raspberry Pi của chúng tôi. Hiện tại, đây là năm Raspberry Pi. Tôi có 8 GB RAM trên chúng, nền tảng là armv7, tôi có 1, 2, 3 người quản lý và sau đó tôi có 1, 2 công nhân. Vì vậy, như bạn có thể thấy ở đây, việc lắp đặt nhiều máy bằng kim loại trần là một cách tuyệt vời khác để tìm hiểu về **Docker** **Swarm**. Tất cả những gì tôi phải làm là cài đặt **Docker**, bạn đã xem tôi thực hiện **Swarm** trong đó chỉ trong vài phút và tôi thậm chí còn triển khai một tệp ngăn xếp với một dịch vụ trong vài phút nữa, miễn cưỡng có sự cố và khắc phục sự cố và mọi thứ trong một khoảng thời gian rất ngắn. Đó là một nền tảng rất thuận tiện để làm việc khi nhóm nhiều máy lại với nhau. Và sau đó, thậm chí còn hơn thế nữa, khi bạn nghĩ về thực tế là bạn sử dụng định dạng giống như Soạn thư, trên thực tế là cùng một định dạng Soạn thư, với một số bổ sung cho mục đích triển khai, để chỉ ném các vùng chứa vào cụm này và không thực sự quan tâm chúng ở đâu hạ cánh, miễn là chúng hạ cánh ở một nơi nào đó mà chúng có thể chạy. Nó giống như thể chúng ta dán năm máy riêng biệt lại với nhau và nhận tất cả sức mạnh trong một, và sau đó tốt hơn nữa, bởi vì chúng ta đang sử dụng **Docker**, chúng ta có khả năng sau đó chia nhỏ sức mạnh đó lên một cấp độ volume như thể nó chỉ là một nhóm các quy trình chạy trên một máy mà chúng ta có thể cách ly hoàn toàn với nhau. Ngay cả khi có nhiều vùng chứa hoạt động cùng nhau trong một ngăn xếp trên nhiều node, chúng tôi có thể giữ chúng cách ly với một ngăn xếp khác chạy trên cùng các node hoặc các node khác nhau.

Định cấu hình Dịch vụ được mở rộng quy mô thông qua Tệp ngăn xếp - echo-replica.yml

Đối với bộ ví dụ tiếp theo này, tôi sẽ sử dụng cục bộ các máy ảo đang chạy của mình, sau đó tôi muốn chuyển đến thư mục example / stacks và mở ngăn xếp ví dụ echo. Bên trong thư mục echo này, có hai tệp YAML riêng biệt. Chúng tôi có toàn cầu và chúng tôi có bản sao. Trước tiên, chúng ta sẽ bắt đầu với tệp bản sao. Vì vậy, tôi đã mở tệp trong trình chỉnh sửa của mình ở đây. Hãy nói về các chi tiết cụ thể. Tôi có một image ở đây. Đây là image mà tôi đã xây dựng với một số điểm cuối bên trong nó cho mục đích học tập, một trong số đó là điểm cuối tiếng vọng, nếu bạn muốn. Tôi chỉ dự định gửi lại thông tin yêu cầu ít nhiều để bạn có thể bắt đầu hiểu cách yêu cầu đến từ bạn, vào cụm và sau đó được chuyển đến một vùng chứa thực tế ở đâu đó để xử lý và sau đó nó quay trở lại như thế nào . Vì vậy, xem loại thông tin tôi có thể thu thập trong quá trình thực hiện và sau đó gửi lại thông tin đó để chúng ta có thể so sánh nó với những gì chúng ta có thể mong đợi so với những gì nó thực sự là. Chúng tôi đang xuất bản cổng 9090 ánh xạ tới ứng dụng web ở đây trên cổng 80. Tôi đang lập bản đồ trong ổ cắm **Docker**. Tôi có một số ví dụ trong đó tôi đọc một số dữ liệu từ API **Docker** giống như Trình trực quan hóa sử dụng nó để thu thập thông tin để hiển thị. Sau đó, bên dưới phần tử triển khai, tôi đã đặt chế độ để nhân rộng và điều này là dư thừa. Nó không cần thiết vì nó muốn là mặc định, nhưng tôi thấy nó được liệt kê rõ ràng ở đây vì đây sẽ là điểm khác biệt lớn so với phiên bản toàn cầu của dịch vụ này. Và sau đó tôi cũng đã đặt một số bản sao là 2. Vì vậy, khi lần đầu tiên chúng ta tạo ngăn xếp, chúng ta sẽ có bao nhiêu vùng chứa trên cụm của mình? Chà, chính xác là 2 vì chúng tôi muốn có 2 bản sao.

Sử dụng tác vụ được tạo mẫu để nối {{.Node.Hostname}} - {{. Task.Slot}} vào tên máy chủ vùng chứa dịch vụ

Và điều cuối cùng tôi đang làm ở đây là sử dụng một mẫu cho tên máy chủ cho vùng chứa, đặt nó thành tiếng vọng - bản sao và sau đó - và sau đó tôi đưa các giá trị vào đây, tên máy chủ của node và sau đó là Tác vụ. . Bạn sẽ thấy lý do tại sao điều này hữu ích sau này. Nếu bạn mở File Explorer, có một tệp README bên trong mà tôi có liên kết đến một chút thông tin khác khi nói đến chính xác những phần giữ chỗ bạn có sẵn để đưa vào bên trong các mẫu và nơi có thể áp dụng các mẫu. Vì vậy, chúng ta có thể sử dụng chúng trong một tên máy chủ, trong một gắn kết và một biến môi trường. Và đây là những trình giữ chỗ khác nhau mà chúng tôi có, và chúng đúng hơn là đang kể về những gì chúng tiêm vào. Và sau đó ở bên dưới, bạn sẽ có một ví dụ về việc này với dịch vụ **docker** tạo và đặt tên máy chủ ở đây. Tất cả chúng đều được hoãn lại cho đến thời điểm mà tác vụ và vùng chứa được tạo, tại thời điểm đó chúng được nội suy và đưa vào các giá trị thực của tác vụ đó đang ở đâu. Và do đó, tác dụng phụ tuyệt vời của việc này là nhiệm vụ của chúng ta sẽ có một tên bao gồm tên máy chủ của node của nó để chúng ta biết nó đang chạy ở đâu.

Triển khai ngăn xếp dựa trên weshigbee / **Swarm**gs2-echo

Được rồi, vậy bạn nghĩ chúng ta sẽ triển khai ngăn xếp bản sao này như thế nào? Chúng ta chỉ cần triển khai ngăn xếp **docker**, chuyển vào tệp ngăn xếp, đặt bản sao ở đó và sau đó, tất nhiên, chúng ta cần đặt tên cho nó. Hãy gọi đây là echor. Đó là tất cả những gì chúng ta phải làm sau đó. Bắn đi. Nếu muốn, chúng ta có thể chuyển sang trình duyệt và xem trình hiển thị, và bạn có thể thấy chúng ta có hai bản sao đã được tạo. Chúng vẫn chưa chạy. Đó là ý nghĩa của chấm đỏ. Có lẽ họ đang lấy một image ngay bây giờ. Bạn có thể thấy trạng thái đang chuẩn bị ở dưới cùng, sau đó bắt đầu và hiện đang chạy. Vì vậy, cả hai đều đang hoạt động ngay bây giờ. Vì vậy, họ phải kéo image xuống. Không thể chắc chắn. họ đã có image trên đó, sau đó họ sẽ khởi động nhanh hơn rất nhiều. Vì vậy, chúng ta có thể thấy rằng chúng ta đã được xếp vào công nhân 2 và công nhân 4. May mắn ngẫu nhiên. Vì vậy, đó cũng là những gì mà việc mở rộng quy mô cũng làm tăng số lượng các trường hợp vùng chứa mà chúng tôi có sẵn để xử lý các yêu cầu đến với dịch vụ đó. Trong trường hợp này, đối với cổng đã xuất bản 9090 đó, vậy tại sao chúng ta không xem xét nó? Vì vậy, tôi đã mở một tab mới, điền trước URL đến 99,101 và chúng ta có thể đạt 9090. Và bạn xem. Chúng tôi đã sử dụng dịch vụ tiếng vọng mới của mình. Đây chỉ là một trong số các điểm cuối có sẵn trong image này mà tôi muốn coi như một image chẩn đoán để tìm hiểu thêm một chút về **Docker**.

echo's / container Điểm cuối để liệt kê các vùng chứa của **Docker** Host của nó - Tương tự như **docker** container ps -a

Một điểm cuối hữu ích khác là điểm cuối vùng chứa. Điều này xảy ra để đọc thông tin về máy chủ **Docker** cụ thể đó. Đầu ra rất giống với lệnh ps container của **docker** với tất cả các cờ, vì vậy các vùng chứa dừng cũng sẽ được hiển thị. Đó là lý do tại sao tôi đã ánh xạ trong **Docker** socket. Nó đọc các vùng chứa đang chạy ở đó. Vì vậy, bạn có thể thấy, trên w2, có vẻ như chúng tôi đã có một phiên bản cũ đã thoát khỏi viz, vì vậy phiên bản này đã thoát cách đây 11 giờ và chúng tôi có 1 tác vụ đang chạy ngay bây giờ cho dịch vụ echo của chúng tôi. Và tất nhiên, điều đó có ý nghĩa. Phải có một vùng chứa đang chạy trên máy chủ này để trả lời yêu cầu của chúng tôi. Theo nghĩa đen, đây là vùng chứa gửi lại cho chúng ta những gì về cơ bản là ps vùng chứa **docker** cho máy chủ cụ thể đó, bao gồm thông tin về chính nó sẽ hiển thị trong ps container của **docker**. Bây giờ, điều thuận tiện, bởi vì tôi sử dụng mẫu đó, bạn có thể thấy w2 ở đây bên trong tên máy chủ, vì đó là tên máy chủ của ứng dụng thực tế đang chạy bên trong vùng chứa đó và nhờ tôi đã đặt nó thành tên máy chủ của node , chúng ta có thể xem node này đang chạy trên node nào, cũng như chúng ta có thể thấy vị trí là số 2 ở đây, vì vậy đây là node thứ hai trong số 2 vùng chứa. Bây giờ, tôi có thể làm mới cái này lặp đi lặp lại, và tôi sẽ tiếp tục lấy w2 - 2. Hãy nói về lý do tại sao lại như vậy, tiếp theo.

Buộc trình duyệt thiết lập các kết nối mới để điều tra các hàm ý định tuyến

Được rồi, vì vậy tôi đã đề cập rằng tôi có thể làm mới trang này có thể cho đến khi tôi xanh mặt và nó có thể sẽ không thay đổi vùng chứa nào trong số hai vùng chứa đang xử lý yêu cầu của tôi, mặc dù có hai trong số đó có thể xử lý lời yêu cầu. Điều đó không phải lúc nào cũng ổn. Nhưng có điều với các trình duyệt, chúng có xu hướng giữ kết nối bằng cách sử dụng keep - living, vì vậy chúng giữ kết nối mở để giúp dễ dàng thực hiện các yêu cầu tiếp theo cho các tài nguyên tương tự hoặc tương tự trên máy chủ. Và do đó, thông tin kết nối cũng không thay đổi. Trong chế độ xem này, bạn có thể thấy thỉnh thoảng có lẽ cổng thay đổi. Và nếu bạn thực sự may mắn, bạn có thể thấy địa chỉ IP thay đổi, cho thấy rằng bạn đã chuyển vùng chứa đang xử lý yêu cầu của mình. Tuy nhiên, nó không thường xuyên và không mấy khi nếu bạn đang làm mới kết nối này, bạn sẽ thấy điều này xảy ra. Thời gian chờ chỉ xảy ra sau một khoảng thời gian không hoạt động hoặc nếu bạn buộc chúng xảy ra. Vì vậy, tôi có một bộ cửa sổ khác được mở bằng Chrome ở đây, sẵn sàng để cho bạn thấy ý tôi. Và điều đầu tiên tôi muốn chỉ ra là bạn có thể thấy ở đây địa chỉ IP cho IP cục bộ, đó sẽ là máy chủ xử lý yêu cầu. Nó khác nhau. Nó không phải là 10.0.0.28. Đó là 10.0.0.27, vì vậy đây là vùng chứa khác của chúng tôi xử lý yêu cầu của chúng tôi. Và nó chỉ xảy ra như vậy, chúng tôi đã may mắn có được một cái đó, và nó khác, nhưng nó sẽ giữ nguyên. Cũng giống như với trình duyệt khác của chúng tôi, cho đến khi bạn xanh mặt, có thể bạn sẽ không may mắn thay đổi IP đó trừ khi, tất nhiên, bạn buộc phải làm như vậy. Vì vậy, trong Chrome, có một số công cụ mà bạn có thể sử dụng để đóng các ổ cắm không hoạt động và xả các nhóm ổ cắm để phá vỡ cơ bản các kết nối đang hoạt động, như nó nói ở đây. Nghe có vẻ như nó có thể phá vỡ các kết nối đang hoạt động, điều này thật tuyệt với tôi. Vì vậy, những gì tôi tìm thấy là bạn có thể nhấp vào node này tại đây, sau đó truy cập vào đây, và đôi khi bạn sẽ gặp may mắn. Khi kết nối đó bị hỏng và bạn làm mới, bạn sẽ nhận được một địa chỉ IP khác cho IP cục bộ, cho biết rằng có thứ khác đang xử lý yêu cầu của bạn, một trong những vùng chứa khác. Nhưng một lần nữa, nếu bạn tiếp tục làm mới ở đây, sẽ không có gì xảy ra. Và tôi thấy rằng với Chrome, bạn có thể tiếp tục làm mới và chỉ cần nhấp vào node này ở phía bên tay phải và bạn sẽ thấy rằng đôi khi nó thay đổi và vì một số lý do, nó không thay đổi ngữ cảnh của cửa sổ khi tôi đang nhấp vào node này ở đây. Vì vậy, nếu bạn đang làm mới thực sự nhanh chóng ở phía bên trái, khi bạn nhấp vào node đó ở bên phải, nó dường như không đánh cắp tiêu điểm. Vì vậy, đó là một cách thực sự tuyệt vời chỉ để xem các giá trị chuyển đổi qua lại khi bạn loại bỏ nhóm ổ cắm. Và bạn đi 27, 28, 27, 28, 27, 28, duh, duh, duh. Rất có thể, những gì bạn sẽ thấy xảy ra là cổng từ xa có thể sẽ thay đổi nếu địa chỉ IP không thay đổi. Điều đó có nghĩa là bạn có một kết nối mới đến với mọi thứ. ID kết nối này cũng sẽ thay đổi, nhưng bạn sẽ không thấy thay đổi IP từ xa và bạn sẽ không thấy thay đổi IP này vì chúng tôi không điều chỉnh thanh địa chỉ của trình duyệt. Và sau đó, đây là một proxy ít nhiều đưa yêu cầu đến của chúng tôi đến 99.101 trên cổng 9090 và cân bằng tải nó vào một trong các vùng chứa có sẵn. Sau đó, nó quay trở lại thông qua địa chỉ này. Đó là lý do tại sao chúng ta có thể thấy rằng từ góc độ của trình xử lý, nó giống như chúng ta là 10.0.0.2, nhưng đó không phải là chúng ta. Đó là ai đó khác thay mặt chúng tôi, bộ cân bằng tải xâm nhập xử lý các yêu cầu gửi đến cụm của bạn và sau đó định tuyến chúng đến vị trí thích hợp. Trong trường hợp này, bao gồm cả với cân bằng tải và cụ thể là với Linux, **Docker** đang thiết lập cái được gọi là máy chủ ảo sử dụng IPVS, do đó, có một IP ảo và đó là những gì đảm nhận việc cân bằng tải cho chúng tôi. Đó là một chức năng cấp hạt nhân . Và tôi không cần phải đi vào chi tiết cụ thể của nó. Bạn có thể tra cứu Netfilter và iptables nếu muốn tìm hiểu thêm và cụ thể là IPVS. Và đừng hiểu nhầm, các chi tiết cụ thể về cách thức hoạt động của IPVS rất hấp dẫn. Bạn nên xem xét nó nếu bạn tò mò, thậm chí xem xét cách nó được triển khai với **Docker**, nhưng hãy nghĩ về bức tranh lớn khi nói đến **Docker** **Swarm**. Đây là một phần khác của hạt nhân mà **Docker**, trong trường hợp này là **Docker** **Swarm**, đang tận dụng lợi thế của việc kết hợp với tất cả các tính năng khác mà nó sử dụng để cô lập vùng chứa, hạn chế tài nguyên, v.v. vào khái niệm này không chỉ là một vùng chứa. , nhưng khái niệm ảo này về mạng lớp phủ với khám phá dịch vụ được cân bằng tải, trong trường hợp này. Được rồi, đưa ngăn xếp ra khỏi cuộc thảo luận đó vào vị trí chúng ta đang ở, điều quan trọng là nếu bạn đang làm việc với trình duyệt, hãy duy trì - tồn tại sẽ ngăn chặn sự thay đổi về người đang xử lý yêu cầu khi bạn có dịch vụ được mở rộng bởi vì kết nối vẫn mở.

Triển khai để kéo image mới nhất

Mặt khác, nếu bạn truy cập vào dòng lệnh, các giá trị bị cắt xén bất thường và chúng đang thay đổi, nhưng tôi nhận ra rằng khi tôi đặt một số định dạng đẹp trên các trang này, tôi đã quên mất rằng tôi muốn thực sự xem xét điều này từ dòng lệnh. Vì vậy, tôi đã thực sự cập nhật image để loại bỏ định dạng bảng trên một vài điểm cuối này bên trong image tiếng vang nhỏ này, để chúng tôi có đầu ra văn bản mà chúng tôi có thể xem trên dòng lệnh và không đi, cái gì, với tất cả các thẻ HTML này. Vì vậy, những gì tôi muốn làm với điều đó là chạy bản cập nhật và triển khai image mới đó. Tất cả những gì tôi đã làm là tạo ra một image mới trên thẻ mới nhất. Bây giờ thông thường, bạn có thể sẽ có một thẻ được cập nhật. Bạn đi vào tệp ngăn xếp của mình và chỉnh sửa thẻ và sau đó triển khai mọi thứ. Chúng tôi không có điều đó bởi vì tôi chỉ đang cập nhật cùng một thẻ, vì vậy những gì chúng tôi có thể làm sau đó, tốt, bạn cho tôi biết. Chúng tôi có thể làm gì để kích hoạt image mới được kéo và các vùng chứa của chúng tôi được tạo lại với nó? Vâng, trước tiên, tôi cần đảm bảo rằng tôi đang ngồi bên cạnh các tệp ngăn xếp của mình và sau đó tất cả những gì tôi phải làm là chạy lại triển khai ngăn xếp **docker** của mình, chỉ định các giá trị chính xác giống nhau, không thay đổi bất kỳ thứ gì bên trong đó chồng tệp. Hãy chờ một giây để chạy, truy cập trình duyệt và tôi có thể làm mới ở đây, và cuối cùng bạn sẽ thấy phiên bản cập nhật của ứng dụng. Của bạn đây. Bạn có thể thấy sự thay đổi đó ở ngay đằng kia không? Vì vậy, bây giờ yêu cầu của chúng tôi đang được xử lý bởi 10.0.0.29 và nếu chúng tôi xóa các ổ cắm của mình, bạn có thể sử dụng, 10.0.0.30 là ổ cắm còn lại xử lý các yêu cầu của chúng tôi. Vì vậy, mọi thứ đã được cập nhật. Ồ, nhân tiện, xem này, chúng ta có w3 ngay ở đây, cái mới. Đó là trên w2 và w4. Và hãy xem liệu chúng ta có thể lấy vùng chứa khác và node của nó để hiển thị ở đây không. Chúng tôi bắt đầu, w1 và w3. Được rồi, hãy chuyển đến Trình hiển thị của chúng tôi và xác nhận điều đó. Và bạn có thể thấy đó thực sự là trường hợp. Trên w1 và w3, bây giờ chúng ta có hai tác vụ hoặc vùng chứa của chúng ta, kéo image mới nhất mà tôi vừa đẩy ra trở lại định dạng văn bản cho một số điểm cuối mà tôi muốn sử dụng từ dòng lệnh.

Quan sát định tuyến bằng Curl và không sử dụng lại kết nối

Bởi vì những gì tôi muốn làm là đến đây với dòng lệnh và chạy curl một lần nữa. Yêu cầu tương tự mà chúng tôi đang thực hiện trong trình duyệt và điều đó trông dễ hiểu hơn rất nhiều. Để thấy lợi ích của sự khác biệt giữa lưu giữ với curl, bạn nghĩ tôi sẽ làm gì ở đây tại dòng lệnh? Hãy kéo cuộc gọi cuộn tròn đó lên. Hãy đặt một chiếc đồng hồ ở đây. Hãy chạy điều này với sự khác biệt và hãy chạy nó sau mỗi 1 giây. Tôi sẽ bỏ tiêu đề để tiết kiệm một số không gian. Bây giờ chúng ta có lẽ không nên in quá nhiều thông tin, vì vậy tôi sẽ đặt lá cờ im lặng ở đây, cũng như viết hoa chữ S để chỉ ra lỗi nếu chúng xuất hiện. Được rồi, điều đó tốt hơn nhiều. Vì vậy, bây giờ chúng tôi đã có đầu ra tương tự như trước đây và bạn có thể thấy mỗi khi có yêu cầu ở đây, chúng tôi hầu như đều nhận được một địa chỉ IP mới. Chúng tôi đã có một cổng từ xa mới, chúng tôi có một ID kết nối mới, có nghĩa là không có lưu trữ tại chỗ và các kết nối đang được tạo mỗi khi một cuộc gọi được thực hiện. Vì vậy, tôi tin rằng điều này sẽ thú vị hơn một chút khi xem xét vì bạn có thể thấy mọi thứ thay đổi khi các yêu cầu được gửi đi và được chuyển đến các trình xử lý dịch vụ khác nhau, đó là các bộ chứa của chúng tôi và chúng tôi có hai trong số chúng ngay bây giờ. Vì vậy, chúng sẽ luân phiên qua lại, có lẽ thỉnh thoảng bạn sẽ thấy cùng một địa chỉ IP hiển thị một lần liên tiếp, nhưng mọi thứ sẽ được cân bằng tải giữa cả hai. Bạn thậm chí có thể thấy ở đó trong tên máy chủ mà bạn có w3, w1 cũng đang thay đổi. Mạng 10.0.0 này là mạng xâm nhập của bạn. Nó có thể được tùy chỉnh, bạn sẽ muốn tra cứu điều đó trong tài liệu nếu bạn có bất kỳ xung đột nào với việc sử dụng dải địa chỉ IP cụ thể này. Và nó để xử lý tất cả lưu lượng truy cập mà bạn có yêu cầu đến các cổng đã xuất bản đi đến các volume sau đó có thể ở bất kỳ đâu trên cụm. Tôi muốn chỉ ra một phần thông tin khác được in ra ở đây. Ở dưới cùng là danh sách các giao diện. Đó là những giao diện quay trở lại từ máy chủ hoặc vùng chứa đã phản hồi yêu cầu. Chỉ cần bạn tò mò, bạn có thể thấy các mạng khác nhau mà vùng chứa đó được kết nối với. Ví dụ: mạng 10.0.0 và tất nhiên, địa chỉ IP của nó là 29, khớp với 29 mà chúng ta thấy ở trên dưới IP cục bộ đáp ứng yêu cầu. Chúng ta cũng có thể thấy mạng 10.0.6. Và cũng có một mạng 172. Mạng 172, liên quan đến việc định tuyến lưu lượng truy cập ra internet. Đó là một mạng cầu nối được cài đặt như một phần của việc thiết lập một bầy, vì vậy khi bạn khởi tạo một bầy. Bạn sẽ thấy **docker**\_gwbridge nếu bạn liệt kê các mạng của mình và điều đó chỉ cung cấp kết nối bên ngoài NAT và chỉ ra thế giới bên ngoài, không phải với các vùng chứa khác. Và, tất nhiên, một giao diện lặp lại. Tất cả các giao diện này nhắc nhở tôi rằng mọi thứ **Docker** thiết lập cho chúng ta dễ dàng như thế nào. Đây là mạng phức tạp điên rồ ngay tại đây chỉ dành cho một vùng chứa, chưa nói đến cho tất cả chúng. Thật sự rất tuyệt khi có thể quét tất cả cấu hình này dưới tấm thảm, mô tả một cách công khai các mạng mà chúng ta cần hoặc ngầm định trong một số trường hợp và chỉ được hưởng lợi từ cấu hình đơn giản đó.

Vai trò của Mạng lớp phủ Ingress

Tôi muốn dành một chút thời gian và tóm tắt lại một chút và chuẩn bị cho chúng ta một phân tích tốt đẹp trong mô-đun tiếp theo về công việc. Vì vậy, chúng tôi đã có một cụm máy ảo ảo mà chúng tôi đang làm việc với các ví dụ gần đây nhất. Tôi sẽ sắp xếp mọi thứ một cách tuyến tính ở đây để tôi có thể nói một chút về mạng và truyền đạt một số khía cạnh nâng cao hơn. Nếu không, tôi e rằng điều này sẽ trở nên hơi khó hiểu. Vì vậy, trong một vài ví dụ cuối cùng, chúng tôi đã xem xét các tình huống trong đó, từ thế giới bên ngoài, bất kể đó là gì, nó có thể chỉ là một máy tính khác trên mạng của bạn hoặc nó có thể ở đâu đó trên internet, ai đó đã gửi yêu cầu đến một trong số các node của bạn. Các node của bạn có địa chỉ IP, họ có thể liên hệ với những địa chỉ đó và họ đã liên hệ với một số cổng đã xuất bản trên các node đó, trong trường hợp của chúng tôi, các cổng đã xuất bản cho dịch vụ echor\_echo này cho ngăn xếp echor, về cơ bản chỉ là image echo đó và tất nhiên, 9090 có sẵn trên toàn cụm nhờ thiết lập cổng đã xuất bản đó và nó ánh xạ vào cổng 80 trên echo. Bây giờ đối với các ví dụ tôi đã làm, tôi luôn đánh cùng một node, nhưng nó không thực sự quan trọng. Bạn có thể đã đạt được bất kỳ node nào. Hãy dành một chút thời gian để cho bạn thấy rằng khi bạn đưa ra yêu cầu, chúng sẽ được chuyển ngẫu nhiên đến các phần phụ trợ khác nhau ở phía sau và đó là lý do tại sao chúng tôi thấy địa chỉ IP thay đổi trong trình duyệt khi chúng tôi buộc đặt lại kết nối và đặc biệt là với curl at dòng lệnh nơi chúng tôi có một kết nối mới mỗi lần. Yêu cầu của chúng tôi về việc được cân bằng tải trên các volume khác nhau mà chúng tôi đang chạy cho dịch vụ của mình nhờ bộ cân bằng tải bầy đàn. Đôi khi đây còn được gọi là lưới định tuyến. Bạn thậm chí có thể nghe thấy từ xâm nhập và điều đó quan trọng bởi vì những gì chúng tôi đã phần nào che đậy là giao tiếp đã diễn ra qua cái được gọi là mạng xâm nhập.

Giao tiếp nội bộ qua Mạng lớp phủ mặc định và tùy chỉnh

Bây giờ tôi đã bỏ một trong những kết nối này ở phía bên trái ở đây, vì vậy tôi muốn giả vờ như chúng ta đang nói về một kết nối. Và tôi muốn đặt câu hỏi, điều gì sẽ xảy ra nếu, khi chúng tôi đưa ra yêu cầu này, chúng tôi cần vùng chứa của chúng tôi đang xử lý yêu cầu để thực hiện một số yêu cầu tiếp theo đối với các dịch vụ khác? Chúng tôi đã nói về vấn đề này như là vấn đề khám phá dịch vụ. Ví dụ, nó thực sự có thể đưa ra một yêu cầu cho chính nó đang chạy trên một node khác. Chính xác thì lưu lượng đó sẽ được định tuyến ở đâu? Chà, có rất nhiều câu trả lời có thể cho câu hỏi đó, tất cả đều xoay quanh thực tế là còn có các mạng khác liên quan ngoài mạng xâm nhập. Mạng xâm nhập dành cho các cổng đã xuất bản. Nó không dành cho giao tiếp chung giữa các vùng chứa và do đó, tại sao khi chúng ta tạo một ngăn xếp, chúng ta lại tạo mạng. Trong trường hợp của chúng tôi, khi chúng tôi tạo ngăn xếp echor, chúng tôi đã tạo một mạng mặc định và điều đó cho phép giao tiếp, sau đó, giữa các vùng chứa được xoay lên trên ngăn xếp echor. Mạng này là mạng ngầm, giống như **Docker** Compose tạo ra các mạng ngầm cho bạn cho dự án của bạn, vì vậy bạn không thấy nó ở bất kỳ đâu được xác định bên trong tệp Soạn của chúng tôi. Bây giờ để hiểu rõ hơn về mạng này, tôi có một ví dụ không chỉ kết nối với một vùng chứa khác. Giả sử chúng tôi tạo ra các phiên bản vùng chứa của chúng tôi trên tất cả các node của chúng tôi. Chúng ta có thể xem xét một loại dịch vụ toàn cầu trong đó một vùng chứa chạy theo nghĩa đen trên mỗi node, trái ngược với việc cung cấp một số bản sao được trải rộng trên cụm. Và ví dụ tôi có sẽ gửi một yêu cầu từ một trong các vùng chứa đến tất cả các vùng chứa còn lại trên ngăn xếp. Vì vậy, điều này đang xảy ra ở hậu trường trên mạng echor mặc định đó và phản hồi sẽ trở lại với chúng tôi thông qua cổng đã xuất bản đó, nhờ vào mạng xâm nhập. Và điều quan trọng cần lưu ý là đây chỉ là một ví dụ đơn giản, vì vậy tôi chỉ phải chạy một image vùng chứa. Bên trong nó, nó có một yêu cầu đặc biệt có thể tiếp cận với các volume khác. Và tôi sẽ chỉ cho bạn cách chúng tôi có thể cắm một miền sao cho nó sẽ tiếp cận với tất cả các vùng chứa lân cận của nó. Không thực sự quan trọng, mặc dù. Nó có thể tiếp cận một vùng chứa trên mạng của nó hoặc tất cả chúng. Những ví dụ này sẽ đủ để cho thấy rằng chúng tôi có thông tin liên lạc đang diễn ra ở hậu trường trên các mạng khác. Chúng tôi thậm chí có thể có một mạng khác, có lẽ là một ứng dụng khác, và có lẽ chúng tôi có thể giao tiếp qua mạng đó cũng như với một số dịch vụ back - end bí mật hoặc một số dịch vụ back - end nhạy cảm và đưa ra các yêu cầu đối với dữ liệu được bảo vệ và bao gồm cả điều đó. Vì vậy, nó không nhất thiết phải là các vùng chứa khác trên cùng một ngăn xếp. Nó có thể là các volume khác trên một ngăn xếp khác. Và đó là một số điều chúng ta sẽ thấy khi chuyển sang khái niệm công việc này. Vì vậy, với tất cả những gì đã nói, hãy sao lưu ở đây để xem điều này trông như thế nào cho yêu cầu này mà tôi sẽ thực hiện trên toàn bộ cụm. Hãy ghi nhớ image này. Ngoài ra, hãy ghi nhớ image của một vùng chứa duy nhất.

Thực hiện Tra cứu DNS từ Một Vùng chứa đến Dịch vụ Riêng của Nó - và Chúng tôi Tìm thấy một Mạng Ngăn xếp Ngầm, Mới!

Được rồi, vì vậy bản demo đầu tiên tôi muốn thực hiện, và nhân tiện, tôi có một trình hiển thị trực quan ở đây ở phía dưới với trạng thái của cụm máy ảo của tôi. Thật thú vị, vì máy ảo của tôi được khởi động lại lần cuối, cả hai tác vụ cho dịch vụ echo của chúng tôi, cả hai đều hạ cánh trên cùng một node, và đó có thể là do N - 1 khởi động nhanh hơn bất kỳ ai khác, và sau đó chỉ cần thực hiện tất cả công việc. Dù sao, hãy ghi nhớ, đó là nơi họ đang ở ngay bây giờ. Ở trên cùng ở đây, tôi cũng có một yêu cầu được đưa vào image này. Nó ít nhiều giống như thực hiện một nslookup hoặc một yêu cầu đào để lấy một tên miền, vì vậy nó là / dns / lookup và tôi sẽ dán thông số này vào đây, domain = echor\_echo, đó là dịch vụ của chúng tôi. Vì vậy, những gì tôi đang làm ở đây là tôi thực sự đi vào vùng chứa này, điều khiển một người xử lý nó, một trong hai người và tôi yêu cầu nó thực hiện tra cứu trên một miền và trả lại cho tôi địa chỉ IP. Địa chỉ IP nào bạn thấy để nói chuyện nếu bạn muốn giao tiếp với bất kỳ vùng chứa nào là một phần của dịch vụ echo này? Bạn có muốn đoán xem bạn có thể thấy gì khi tôi chạy nó không? Vì vậy, chúng tôi nhận được một cái gì đó tương tự như dòng lệnh. Tôi đã cắt ngắn đầu ra ở đây chỉ để giữ cho nó đơn giản. Tôi đã thực hiện yêu cầu này với một ứng dụng .NET và sau đó tôi vừa lấy ra địa chỉ IP quay trở lại và tôi đã hiển thị loại địa chỉ IP là IPv4. Bây giờ, điều đó thật thú vị. Đó là 10.0.6.2. Điều đó trông hơi khác một chút. Hãy chia màn hình trên cùng ở đây và ở bên phải, tôi đã đưa ra yêu cầu mà chúng tôi đã đưa ra ban đầu khi chúng tôi xem xét kết nối liên tục giữa các trình xử lý và chúng tôi thấy địa chỉ IP này thay đổi ở đây. Bây giờ sự khác biệt là đây là vào ngày 10.0.0.11. Chà, đây là mạng xâm nhập của chúng ta. Vì vậy, yêu cầu của chúng tôi sẽ đi qua mạng xâm nhập trong trường hợp này, bởi vì chúng tôi đang tiếp cận với một cổng đã xuất bản và nó đang được xử lý bởi một trong các vùng chứa của chúng tôi và đôi khi chúng tôi có thể làm mới. Và hãy nhìn xem, bạn có thể thấy chúng tôi có một thay đổi ở đây thành 10.0.0.32, vẫn trên mạng 10.0.0, vì điều đó không ổn giữa việc được phản hồi bởi hai vùng chứa riêng biệt này đang chạy dịch vụ echo của chúng tôi. Tất nhiên, tôi có thể làm mới nó trong một thời gian dài ở đây và không bao giờ thấy thay đổi nữa. Vì vậy, có hai mạng riêng biệt liên quan, 10.0.0 và 10.0.6, đó là những gì tôi muốn bạn biết về điều này. Tôi sẽ phóng to hơn một chút nếu điều đó có ích. Vì vậy, khi chúng tôi giao tiếp với dịch vụ echo của mình, chúng tôi sử dụng mạng xâm nhập từ cổng xuất bản. Khi một trong các vùng chứa của chúng ta cho ngăn xếp của chúng ta muốn giao tiếp với một vùng chứa khác trên ngăn xếp, nó sẽ sử dụng một mạng khác, mạng 10.0.6 này. Và đó chính xác là sự tách biệt mà chúng ta thấy ở đây giữa mạng xâm nhập, hoặc mạng 10.0.0 và những gì chúng ta sẽ thấy ở đây là mạng ECHOR\_DEFAULT tại 10.0.6. Bây giờ, nếu bạn chuyển sang dòng lệnh, chúng tôi có thể xác nhận các mạng này thực sự xếp hàng như tôi đã đề xuất. Chúng tôi có thể chạy một mạng **docker** ls, điều đó cho thấy được kết nối với một trong các node nơi người quản lý đang ở và có lẽ nơi một trong các vùng chứa của bạn đang chạy, bạn sẽ gặp may mắn nhất. Các node công nhân không phải lúc nào cũng có quyền truy cập vào các mạng tùy chỉnh có thể tồn tại như mạng ECHOR\_DEFAULT của chúng tôi, trừ khi tất nhiên có một vùng chứa trên chúng cần nó. Đó là một phần hiệu quả của **Swarm** và đảm bảo rằng nó không mở rộng quy mô mạng quá lớn, đặc biệt là khi kích thước cụm của bạn tăng lên. Dù sao thì, xâm nhập là điều tôi quan tâm lúc này. Những gì tôi có thể làm sau đó là chạy một kiểm tra mạng **docker** và chuyển qua quá trình xâm nhập, và tôi sẽ định dạng nó bằng jq, chỉ để lấy một số màu. Nhìn lên trên cùng ở đây. Bạn sẽ thấy có phần Cấu hình bên dưới mà bạn có Mạng con được đặt thành 10.0.0.0, không có gì ngạc nhiên, / 24. Vì vậy, có mạng 10.0.0 của chúng tôi và bạn có thể thấy cổng vào đó là 10.0.01. Bây giờ nếu chúng ta thực sự tò mò, chúng ta có thể đến đây và nói, được rồi, vì vậy chúng ta có IP từ xa này; nó vẫn là 10.0.0.2 và những thứ này đã thay đổi, nhân tiện, các máy ảo của tôi được sử dụng và ngoại tuyến khi tôi làm việc thông qua các bản demo này, nhưng thật không may, IP này không thay đổi. Nó cũng có thể thay đổi, nhưng nó không, và điều đó thật tuyệt; 10.0.0.2. Chúng tôi có thể xem qua đây và xem liệu chúng tôi có thể tìm thấy 10.0.0.2 ở bất kỳ đâu, và bạn xem, chúng tôi đã có 10.0.0.2. Đây thực sự được liệt kê là hộp cát đi vào hoặc hộp cát, vì vậy đây là nơi lưu lượng truy cập của chúng tôi được định tuyến qua. Đây là proxy của chúng tôi nếu bạn muốn, đó là chuyển các yêu cầu sau đó đến các vùng chứa khác nhau ở hậu trường và chúng tôi chỉ có thể nhìn thấy chúng vì tôi đã rất may mắn nắm bắt được và gửi lại yêu cầu tại đây trong yêu cầu phản hồi. Nếu không, chúng tôi sẽ không biết địa chỉ IP này. Nó thực sự sẽ giống như mọi thứ có thể được xử lý chủ yếu bởi máy chủ yêu cầu cụ thể này. Các IP 10.0.0 sẽ bị che khuất khỏi chúng tôi. Bây giờ, có một chút thông tin bị thiếu trong chế độ xem này. Tham gia cùng tôi trong video tiếp theo cho điều đó.

- Cờ của Kiểm tra mạng **Docker** chứa đầy thông tin chi tiết phong phú về mạng Ingress

Vì vậy, trong khi tôi đang đào sâu vào mạng xâm nhập này bằng cách kiểm tra nó, tôi đã tìm thấy 10.0.0.2 trông giống như một proxy hoặc bộ cân bằng tải để chuyển yêu cầu đến một trong các vùng chứa dịch vụ và chuyển đổi qua lại dưới dạng các yêu cầu mới mời vào. Nhưng tôi không có nhiều thông tin về 10.0.0.2 này chính xác là gì và nó ở đâu và đó là vì thực sự có một cờ ẩn ở đây mà bạn có thể bỏ lỡ nếu không, và để hiển thị cho bạn lá cờ này, hãy xem thư mục ngăn xếp, vào thư mục căng thẳng hoặc bạn thực sự có thể chỉ xem nội dung của nó từ đây và nhìn vào bên trong thư mục ghi chú bên trong đó. Tôi có kiểm tra mạng chi tiết khác này, tôi muốn chạy nó vì vậy tôi sẽ chỉ đường dẫn đến nó, nhưng trước khi chạy nó, tôi sẽ in nó ra cho bạn để bạn có thể thấy điều này làm gì. Vì vậy, chúng tôi sẽ chạy một cái gì đó rất giống với những gì chúng tôi vừa làm với mạng kiểm tra và thâm nhập mạng **docker**, ngoại trừ chúng tôi sẽ khác nó với kiểm tra mạng **docker** và sau đó chúng tôi sẽ thêm một cờ chi tiết, và một lần nữa, cùng một mạng xâm nhập. Bây giờ đây là nhãn tôi đang cho bạn xem sẽ hiển thị ở đầu khác biệt này. Đây là các lệnh thực tế hoàn toàn giống nhau ở đây và thông qua quá trình thay thế ở đây, chúng tôi chuyển điều này sang lệnh icdiff và tôi yêu cầu xem toàn bộ tệp, không chỉ hiển thị những gì đã thay đổi. Vì vậy, chúng ta có thể tiếp tục và chạy nó sau đó và điều này sẽ cho chúng ta thấy nó trông như thế nào khi chúng ta thêm cờ tiết và tôi muốn hiển thị nó cạnh nhau. Được chứ. Tôi nghĩ điều đó tốt ở đó. Bây giờ tôi biết điều này sẽ hơi khó đọc, tôi sẽ đọc cho bạn qua nội dung. Tôi chỉ muốn bạn thấy rằng ở trên đây, chúng ta có khá nhiều kết quả chính xác giống nhau. Màu vàng cũng vậy. Vì vậy, mọi thứ vẫn như cũ cho đến khi chúng ta đi đến điểm này, nơi có một trường dịch vụ và bên dưới đây, chúng ta có thêm thông tin mà nếu không chúng ta sẽ không thấy. Bây giờ tôi biết là rất khó để xem, nhưng bạn có thấy bất kỳ điều gì thú vị bên dưới thuộc tính hoặc trường dịch vụ này không, Chà, bên dưới địa chỉ IP cho các điểm cuối xâm nhập khác nhau trông rất giống với những gì chúng ta vừa xem, chúng ta có 10.0.0.2 một lần nữa, trên thực tế, chúng ta hãy tìm 10.0.0.2 ở đây. Có ví dụ ban đầu mà chúng tôi tìm thấy, nó ở cả hai đầu ra, nhưng sau đó trong đầu ra dài dòng, chúng tôi nhận được một mục bổ sung và cái này có một chút thông tin đính kèm với nó, một trường thông tin có IP máy chủ, huh, nó đã được thiết lập đến 99,101. Vì vậy, có vẻ như bộ cân bằng tải hoặc proxy 10.0.0.2 này, bất cứ điều gì bạn muốn gọi nó, nằm trong một số loại hộp cát điểm cuối xâm nhập, một số loại vùng chứa - khái niệm ish ẩn đằng sau cảnh nằm trên 99.101 và đó có thể là do chúng tôi 'đã thực hiện các yêu cầu của chúng tôi cho đến nay cho đến năm 99.101. Vì vậy, ở phía bên này, nếu bạn nhận được 99.101, chúng tôi nhận được 10.0.0.2. Điều gì xảy ra nếu chúng ta chuyển đến 102, thay vì 101, đó là quản lý 2? Chà, chúng ta có thể đoán trước được điều này. Hãy xuống đây và tìm 102. Được rồi, đây là phần chúng ta đã thấy ở cả hai phía. Đây là một chút thông tin bổ sung. Được rồi, vì vậy 99.102 ánh xạ tới những gì trông giống như 10.0.0.3 trái ngược với 10.0.0.2 trên 99.101 ở đây. Vì vậy, nếu phép toán của tôi là đúng, chúng ta nên lấy lại 10.0.0.3, thay vì 10.0.0.2. Bây giờ chúng ta hãy tìm hiểu xem có phải như vậy không nhé. Vì vậy, hãy xem ngay tại đây. Tôi sẽ chạy yêu cầu với 102 và nhìn vào đó, bạn biết gì không. Chúng tôi đã có 10.0.0.3 đúng như nghi ngờ. So với cờ dài dòng này, bạn thực sự có thể đào sâu vào một số chi tiết đằng sau hậu trường và tôi không có tất cả thời gian trên thế giới để giải thích tất cả những điều này. Tôi chỉ muốn bạn thấy rằng nếu bạn quay qua kiểm tra mạng, bạn sẽ học được rất nhiều và có rất nhiều thứ để học ở đây mà tôi không thể đề cập đến trong khóa học này, nhưng tôi khuyến khích bạn nên xem xét của riêng bạn nếu bạn tò mò về cách mạng hoạt động. Vì vậy, những gì trông giống như mỗi node khác nhau của chúng tôi có một số loại cân bằng tải nằm bên cạnh cổng 9090 đó đã được xuất bản, chính yếu tố nhỏ đó sau đó cân bằng tải các yêu cầu đối với các vùng chứa dịch vụ phụ trợ khác nhau và mỗi cổng được xuất bản là sẽ có bộ cân bằng tải nhỏ của riêng nó.

Điều tra cách hoạt động của mạng lớp phủ echor\_default đối với giao tiếp nội bộ

Được rồi, tôi muốn nói ngắn gọn về khái niệm tương tự với mạng xâm nhập và cân bằng tải liên quan đến mạng echor\_default của chúng tôi. Rốt cuộc, chúng ta đi từ một trong các volume của mình sang volume kia như thế nào? Chỉ cần cung cấp tra cứu DNS hoặc chỉ cần liên hệ với curl và tên dịch vụ của chúng tôi. Làm cách nào để chúng tôi thực hiện điều đó và nhận được phản hồi từ một trong những vùng chứa cung cấp dịch vụ mà chúng tôi quan tâm? Rốt cuộc, có thể có nhiều hơn một trong số chúng. Làm thế nào để chúng tôi biết cái nào để đi đến? Tại sao chúng tôi chỉ nhận được một địa chỉ IP trở lại đây khi tôi thực hiện tra cứu trên echor\_echo? Đó là 10.0.6.2 và nó sẽ luôn ở giá trị đó, tất nhiên trừ khi chúng tôi thay đổi dịch vụ của mình, có thể khởi động lại máy ảo của chúng tôi, giá trị này có thể thay đổi, nhưng nó sẽ không đổi miễn là cụm của chúng tôi đang hoạt động, giống như cách mà IP từ xa của chúng tôi ở đây không đổi miễn là chúng tôi đang đi vào cùng một node. Vì vậy, hãy xem nhanh dòng lệnh ở cùng một điểm khác biệt mà chúng ta đã xem xét kiểm tra mạng với cờ tiết và không có, ngoại trừ lần này tôi đang ngồi bên trong thư mục ghi chú và giữa các bản ghi, tôi đã tiếp tục và tạo một bản sao thứ hai của tập lệnh khác biệt đó và bản sao này dành cho mạng mặc định của echor. Và sau đó tôi tạo tệp gốc, vì vậy nó đã được đổi tên, và sau đó nếu bạn nhìn vào nội dung của chúng, tôi đã thiết lập một khác biệt ở đây, chỉ để cho bạn thấy không có nhiều sự khác biệt giữa các tệp này, tôi đã tiếp tục và đã tham số hóa lệnh dưới đây để đưa vào mạng ở mỗi vị trí khác nhau cần thiết. Đây giống như một chức năng ở đây đã được chia sẻ nơi tôi copypasta - ed nó. Vì vậy, hãy tiếp tục và chạy tập lệnh echor\_default đó và xem xét kiểm tra mạng có và không có cờ tiết. Được rồi, bây giờ nhìn qua đây, bạn sẽ thấy ở phía dưới có một phần bổ sung khi chúng ta sử dụng cờ tiết. Và nhân tiện, các tiêu đề ở đây sẽ hiển thị cho bạn lệnh nào đã được chạy. Điều này được cung cấp bởi các nhãn đối số bên trong tập lệnh đó. Vì vậy, trong trường hợp này, tôi đang thêm cờ tiết. Chà, một cái gì đó sẽ trông thú vị ở đây. Chúng tôi có VIP là 10.0.6.2. Vâng, VIP, đó là những gì IPVS được sử dụng như một IP ảo. Vì vậy, như bạn có thể nghi ngờ, IPVS không chỉ được sử dụng trên mạng xâm nhập, nó còn được sử dụng trên các mạng tùy chỉnh của chúng tôi. Vì vậy, 10.0.6.2, điều đó quen thuộc vì chúng tôi đã thấy điều đó ở đây khi chúng tôi thực hiện tra cứu DNS trên tên dịch vụ của chúng tôi, echor\_echo. Hãy nhớ rằng, đây chỉ là dịch vụ echo của chúng tôi có tiền tố là tên ngăn xếp. Chúng tôi nhận được 10.0.6.2 trở lại. Và đó là một IP ảo. Và có vẻ như nó cung cấp quyền truy cập vào 2 IP riêng biệt là 10.0.6.3 và 10.0.6.13. Và ngay bây giờ, cả hai đều có liên quan trên 99.101 và điều đó chỉ xảy ra vì dịch vụ của chúng tôi có 2 vùng chứa trên cùng một node m1. Đây là một bảng tra cứu, nếu bạn muốn, về cách đi từ IP ảo đến thực tế được gọi là dịch vụ thực đằng sau hậu trường, hoặc máy chủ thực. Nếu bạn thực sự muốn tìm hiểu nhiều, bạn có thể tìm hiểu kỹ hơn một chút giống như chúng tôi đã làm với mạng xâm nhập và chúng tôi có thể thực hiện yêu cầu này từ trình duyệt tại đây để nhận thông tin chi tiết về kết nối, hãy truy cập ngay vào cổng đã xuất bản 9090 đó , nhưng nếu tôi muốn xem mọi thứ trông như thế nào từ quan điểm của một trong những vùng chứa dịch vụ này, dịch vụ echo của tôi, tôi sẽ cần phải vào vùng chứa đó vì tôi không cung cấp bất kỳ loại giao diện web nào cho điều này. Vì vậy, tôi có một cửa sổ mới được mở ra ở đây, tôi đã có một nhà thực thi vùng chứa **Docker** được ‑populated trước, vì vậy tôi sẽ nhảy o, n bên trong, ngữ cảnh của tôi được đặt thành m1 vì tôi cần phải có để tôi có thể truy cập vùng chứa và sau đó tôi đã hoàn thành tab để nhận dịch vụ echor, một trong hai dịch vụ và sau đó tôi sẽ chạy chương trình bash khi thực hiện hoặc đột nhập vào vùng chứa này. Tôi có thể đã làm điều này thay vì sử dụng các giao diện này ngay tại đây, đặc biệt là giao diện này ngay tại đây, bởi vì bây giờ tôi có thể làm một cái gì đó giống như nslookup, và chúng tôi có thể đặt echor\_echo, và bạn có thể thấy chúng tôi cũng lấy lại 10.0.6.2, giống như chúng tôi có ở đây trong trình duyệt. Hoặc tôi có thể sử dụng đào nếu tôi muốn và tôi sẽ nhận lại được kết quả tương tự. Điều này đến từ máy chủ DNS nhúng do **Docker** cung cấp như một phần của các mạng lớp phủ tùy chỉnh này, rất giống với những gì chúng tôi đã có với các mạng tùy chỉnh với **Docker** Compose. Những máy chủ này cũng có máy chủ DNS nhúng, nơi bạn có thể chỉ cần cung cấp tên của bất kỳ dịch vụ nào bạn đang làm việc và nó sẽ ánh xạ nó tới bất kỳ địa chỉ IP nào bạn có thể sử dụng để giao tiếp với nó. Và trong trường hợp này, một mạng lớp phủ, vì chúng ta có thể có nhiều trường hợp của một dịch vụ, đây là một bộ cân bằng tải, hoặc giao diện người dùng sẽ trở thành một bộ cân bằng tải, đưa chúng ta trở lại các vùng chứa thực sự cung cấp dịch vụ này, một trong số đó chúng tôi đang ngồi bên trong ngay bây giờ. Vì vậy, nếu chúng ta thực sự muốn vui chơi ngay bây giờ, chúng ta không chỉ có thể đào bới, chúng ta có thể làm xoăn. Và tôi đã tiếp tục và cài đặt sẵn curl ở đây. Nó không nằm bên trong image, nhưng tôi đã thêm nó với bản cập nhật tiện ích và cuộn cài đặt ứng dụng. Bây giờ, bạn nghĩ chúng ta sẽ thấy gì khi chạy nó ngay tại đây? Vì vậy, hãy nhớ rằng, dịch vụ echo của chúng tôi chạy trên cổng 80. Chúng tôi không muốn chuyển đến 9090 ngay bây giờ, chúng tôi muốn chuyển đến cổng 80 vì chúng tôi đang ở bên trong mạng. Chúng tôi có quyền truy cập vào các cổng tiếp xúc thực tế của ứng dụng của chúng tôi. Khi tôi chạy điều này, tôi sẽ nhận được cùng một đầu ra mà chúng tôi đã có trong trình duyệt khi chúng tôi chạm vào cổng 9090. Chúng tôi sẽ có các địa chỉ IP khác nhau bởi vì chúng tôi sẽ sử dụng một mạng khác để giao tiếp. Vì vậy, tôi yêu cầu nói chuyện với dịch vụ của tôi trên cổng 80, vì vậy tôi đang ở trong mạng mặc định echor tùy chỉnh của mình. Điều này ánh xạ đến 10.0.6.2 như chúng ta đã thấy, và sau đó cuối cùng được ánh xạ thành 10.0.6.14, tải cân bằng thành 10.0.6.13, trong trường hợp này, nếu chúng ta chạy điều này, trớ trêu thay là đủ lần. Nhìn vào đó, 10.0.6.3, chúng tôi đang nhận được phản hồi từ chính chúng tôi và tôi có thể cho bạn thấy điều đó ở đây. Ở đây chúng tôi có danh sách địa chỉ IP 4 về địa chỉ của chúng tôi trên máy này và bạn sẽ thấy ở đây là 10.0.6.3, đó là những gì chúng tôi nhận được vào lần thứ hai chúng tôi thu thập điểm cuối kiểm tra hoặc tiếng vọng. Nếu bạn tò mò, bạn có thể đi qua và xem xét sự khác biệt đó, và bây giờ là 10.0.6.14, trung gian, có vẻ như bộ cân bằng tải của chúng tôi ngay tại đây. Điều đó đến từ đâu? Hãy đến với cửa sổ 6.14 này, và này, nó hiển thị ở đây ở cả hai phía. Điều này trông giống như một bộ cân bằng tải. Nó có lb phía trước và sau đó là tên mạng của chúng tôi, echor\_default. Vì vậy, đây là bộ cân bằng tải của chúng tôi cho mạng của chúng tôi. Vì vậy, chúng tôi có VIP của chúng tôi là 10.0.6.2, đó là những gì giải quyết cho tên dịch vụ. Điều đó được ánh xạ tới bộ cân bằng tải tại 6.14, sau đó định tuyến yêu cầu đến một trong các vùng chứa ở đây, 6.3 hoặc 6.13. Bây giờ, có rất nhiều thông tin phong phú ở đây. Tôi không muốn đi sâu vào vấn đề này nữa, nhưng tôi chỉ muốn cho bạn một cái nhìn sơ lược về những gì đang tiềm ẩn và giúp bạn bắt đầu phân biệt và hiểu những gì đang diễn ra đằng sau hậu trường khi nói đến các mạng tùy chỉnh này, cũng như mạng xâm nhập. Bởi vì tôi thực sự nghĩ rằng sẽ rất hữu ích khi hiểu được, đặc biệt là chúng ta có nhiều mạng lưới tham gia. Nhưng đó là tất cả những gì bạn bỏ đi. Tôi nghĩ bạn đang ở một vị trí tuyệt vời.

dig servicename Trả về VIP của Dịch vụ của chúng tôi - ngược lại - dig task.servicename Trả về Danh sách IP dnsrr của Dịch vụ của Chúng tôi

Được rồi, một lúc trước, tôi đã nói rằng chúng tôi có một yêu cầu đơn giản này mà chúng tôi đang thực hiện để liên hệ với thông tin phản hồi đó hoặc kiểm tra thông tin để xem ai đang xử lý yêu cầu của chúng tôi. Điều đó thật thú vị, nhưng tôi đã nói rằng có một kịch bản khác mà chúng ta có thể có bất kỳ số lượng container nào chạy vượt quá chỉ hai và trong trường hợp của chúng tôi, chúng tôi sẽ có một bộ chứa chạy trên mỗi node. Và sau đó điều gì sẽ xảy ra nếu chúng ta muốn tiếp cận với từng người trong số này, có lẽ để lấy một số thông tin chẩn đoán? Có cách nào đó mà chúng ta có thể vượt ra ngoài bộ cân bằng tải đó, điều này không cho chúng ta nhiều lựa chọn về người xử lý yêu cầu của chúng ta và nhận một số loại danh sách các dịch vụ khác nhau và quyết định xem chúng ta sẽ yêu cầu thông tin từ ai? Vâng, đúng vậy, chúng ta hoàn toàn có thể làm được điều đó. Vì vậy, trong trình duyệt của tôi, thay vì echor\_default và tôi có thể thực hiện việc này tại đây hoặc tôi có thể đi đến chính vùng chứa và thực hiện việc này, có thể làm cả hai để hiển thị cho bạn, tôi có thể chỉ cần thêm yêu cầu này với các tác vụ, vì vậy miền sẽ là các nhiệm vụ.echor\_echo. Trong trường hợp này, tôi đang nói, này, hãy cung cấp cho tôi danh sách tất cả các nhiệm vụ để cung cấp dịch vụ echor\_echo này. Và bây giờ bạn có thể thấy chúng tôi không lấy lại 10.0.6.2. Chúng tôi nhận được 10.0.6.3 và 13, mà chúng tôi biết là hai địa chỉ IP của các vùng chứa, hãy xem chúng ở đây, cung cấp dịch vụ echo. Vì vậy, tôi có thể làm điều đó bằng một cách khác, tôi có thể đến đây và thực hiện cùng một yêu cầu chính xác, task.echor\_echo. Và bây giờ, thay vì 10.0.6.2, tôi nhận được 10.0.6.13 và 10.0.6.3. Đây là truy cập những gì được gọi là phương pháp tiếp cận vòng quay DNS . Trên thực tế, nếu tôi chạy điều này đủ lần, bạn sẽ thấy các địa chỉ IP lật xung quanh đây trong danh sách. Của bạn đây. Bạn có thể thấy rằng 10.0.6.3 và 10.0.6.13. Đó là trường hợp bởi vì sau đó, đôi khi bạn có thể chỉ cần chọn cái đầu tiên quay lại nếu bạn sử dụng phương pháp này và bạn sẽ biết rằng bạn không áp đảo hệ thống bằng cách đi đến chỉ một vùng chứa mỗi lần vì chúng được xoay vòng cho Bạn. Chúng đang bị trộn lẫn khi bạn lấy lại kết quả. Vì vậy, điều này ngược lại, và có lẽ nslookup ở đây rõ ràng hơn một chút. Chúng ta bắt đầu. Vì vậy, điều này trái ngược với việc sử dụng bộ cân bằng tải của chúng tôi ở đây với IP ảo của chúng tôi hoặc chỉ truy cập trực tiếp vào chính các vùng chứa. Đây là phương pháp tiếp cận vòng quay DNS của chúng tôi và đây là phương pháp VIP hoặc IP ảo của chúng tôi với cân bằng tải. Ở trên, chúng tôi phải cung cấp cân bằng tải, nếu bạn muốn, bằng cách chọn một trong các mục nhập quay lại từ phản hồi DNS và hy vọng không phải chọn cùng một mục mỗi lần. Vì vậy, đôi khi mọi người sử dụng cách tiếp cận này ở trên, và sau đó họ sẽ đặt bộ cân bằng tải của riêng họ trước mọi thứ nếu họ không muốn sử dụng bộ cân bằng tải tích hợp sẵn này . Bây giờ về việc giải quyết các tên miền khác nhau này và lấy lại địa chỉ IP, tôi muốn chỉ ra máy chủ DNS đang được tham chiếu ở đây. Vì vậy, điều này được **Docker** cung cấp như một chức năng của các mạng lớp phủ tùy chỉnh này. Nó xử lý DNS cho chúng tôi đằng sau hậu trường, vì vậy chúng tôi có thể biết những tên dịch vụ thân thiện này mà chúng tôi nhập vào các tệp ngăn xếp và nó sẽ xử lý mọi thứ khác để chúng tôi đưa chúng tôi trở thành VIP nếu nó cân bằng tải hoặc nếu đó là vòng lặp , cung cấp cho chúng tôi danh sách các nhiệm vụ đang xử lý các yêu cầu đối với dịch vụ của chúng tôi. Tôi nghĩ điều đó khá mạnh mẽ. Và vì vậy, hãy lưu ý rằng node này đang chạy trên node hiện tại mà tôi đang ở trên và tôi biết điều này được **Docker** thiết lập cụ thể bởi vì thông thường trên mạng của tôi, tôi có máy chủ DNS của riêng mình và nó không được hiển thị ở đây.

Sử dụng dnsrr để khám phá dịch vụ nội bộ và vùng chứa đến vùng chứa trên mạng lớp phủ phụ trợ

Vì vậy, bây giờ cho phần thú vị. Chúng tôi có điểm cuối vùng chứa này sẽ trả về các vùng chứa của bất kỳ vùng chứa dịch vụ nào được chọn ngẫu nhiên phản hồi, vì vậy nó sẽ là tất cả các vùng chứa trên máy chủ của nó. Và thú vị ngay bây giờ, danh sách này sẽ giống nhau bất kể vì cả hai vùng chứa đang chạy trên cùng một máy chủ và tôi sẽ cho bạn thấy điều đó thực sự. Tôi có một điểm cuối khác có sẵn. Ở phía trước của các vùng chứa, hãy đặt cụm và sau đó ở cuối ở đây, đặt vào một chuỗi truy vấn có miền được đặt thành task.echor\_echo. Vì vậy, tôi đã xây dựng một phần chức năng nhỏ cho phép bạn ra lệnh cho một điểm cuối vùng chứa tương tự để truy vấn tất cả các phản hồi quay lại từ yêu cầu DNS này. Vì vậy, ví dụ: nếu chúng tôi lấy lại hai địa chỉ IP cho yêu cầu DNS này, cả hai IP đó sẽ được liên hệ tại điểm cuối / container để nhận danh sách các vùng chứa mà chúng có trên máy chủ của mình và sau đó tất cả các IP đó sẽ được kết hợp với nhau và được đưa trở lại trang này với điểm cuối cụm / vùng chứa mới này. Vì vậy, tôi sẽ gửi cái này ở đây và chúng ta sẽ xem nó trông như thế nào với hai vùng chứa. Bây giờ để làm mới nhanh chóng. Đây là truy vấn tìm kiếm DNS của các task.echor\_echo và bạn có thể thấy nó quay trở lại với hai địa chỉ IP cho hai vùng chứa echo của chúng tôi thay vì nếu chúng tôi chỉ yêu cầu dịch vụ, mà chúng tôi sẽ sử dụng VIP và duy nhất cung cấp cho chúng tôi địa chỉ IP duy nhất để cân bằng tải và do đó, đối với task.ecor\_echo, chúng tôi nhận được hai máy chủ riêng biệt trở lại, điều này xảy ra là cùng một máy chủ lưu trữ, trong trường hợp này, vì hai vùng chứa nằm trên cùng một máy chủ lưu trữ. Trước khi chúng ta tiếp tục, tôi muốn cho bạn thấy cách chúng ta có thể kết hợp mọi thứ ở đây và loại bỏ một số lịch sử có thể đang hiển thị với nhiều thay đổi đối với cụm của chúng ta qua các ví dụ khác nhau. Ví dụ: những danh sách này có thể thực sự dài đối với các vùng chứa đã chết và nếu bạn muốn xóa một số trong số đó và / hoặc chỉ muốn trộn nó lại để bạn có thể xem nhiều hơn chỉ là node m1, thì chúng ta hãy xem lúc mở rộng quy mô tại dòng lệnh. Vì vậy, tôi sẽ chạy điều này ở đây ở quy mô dịch vụ **docker** và tôi sẽ tăng quy mô hai dịch vụ nổi bật của chúng tôi là viz và echor\_echo xuống còn 0. Bạn thực sự có thể chia tỷ lệ nhiều cùng một lúc. Và ngay sau khi điều đó hoàn thành, tôi sẽ chạy quy mô xuống một lần nữa. Tôi đã gặp may mắn hơn với việc xóa bỏ một số trạng thái cũ, nhưng quan trọng nhất, điều này sẽ thực sự làm xáo trộn các volume của chúng tôi và hy vọng đảm bảo rằng chúng hạ cánh trên các node khác nhau và sau đó tôi sẽ quay lại và thu nhỏ chúng trở lại thăng. Vì vậy, thang đo kép xuống 0 dường như đã xóa mọi thứ về lịch sử nhiệm vụ, hãy tạo cơ hội để khởi động, đó là hai dịch vụ tiếng vọng. Và bây giờ nếu tôi thực hiện một ps dịch vụ **docker** ở đây trên viz, chúng tôi chỉ có một mục nhập và hy vọng, chúng tôi cũng có điều tương tự ở đây với echor\_echo. Chúng tôi có hai mục cho cả hai đang chạy ngay bây giờ. Tuyệt quá. Và hiệu quả thực sự ở đây là chúng tôi không có bất kỳ lịch sử nhiệm vụ cũ nào bây giờ. Và nếu bạn nhìn vào cột node, bạn có thể thấy bây giờ chúng tôi có hai node riêng biệt cho dịch vụ echor của chúng tôi và 1 trong w3. Chúng tôi cũng có thể mở trình duyệt để xem mọi thứ đã di chuyển xung quanh và xác nhận rằng chúng tôi có m1 và w3 cho dịch vụ echor\_echo của mình. Bây giờ tôi muốn hỏi bạn một câu hỏi. Chúng tôi sẽ nhận lại được gì khi gửi lại trang này ngay bây giờ. Vâng, đây là nó đi. Đó có phải là những gì bạn đang mong đợi? Mọi thứ bây giờ trông đơn giản hơn một chút. Bây giờ tôi muốn chỉ ra rằng vẫn còn một vùng chứa đã chết được liệt kê ở đây. Đây là trình hiển thị độc lập mà chúng tôi đã triển khai. Rõ ràng, chúng tôi sẽ không giải quyết vấn đề đó bằng cách xử lý các giao diện **Docker** **Swarm** vì điều đó được tạo bằng các giao diện **Docker** độc lập. Chúng tôi có thể lược bỏ mọi thứ trên các node khác nhau của mình nếu chúng tôi muốn xóa một số lịch sử độc lập. Và một lợi ích khác của việc làm này, bây giờ bạn có thể thấy chúng tôi có w3 là 1 trong các node mà chúng tôi đang làm việc và m1, vì vậy bây giờ bạn có thể thấy các vùng chứa từ hai máy chủ riêng biệt này và chúng khác nhau. Vì vậy, bây giờ chỉ cần tưởng tượng nếu chúng ta có cái này cho tất cả các node của chúng ta và để hiểu rõ hơn về điều này, chỉ cần nói rõ, khi yêu cầu đó đến một trong các vùng chứa, ngay bây giờ chúng ta chỉ có hai, có thể có nhiều trong số chúng, nó sau đó tiếp cận các vùng chứa khác nhau tùy thuộc vào miền mà chúng tôi cung cấp và gọi về cơ bản là điểm cuối của vùng chứa trên các vùng chứa đó, kết hợp các kết quả sao lưu và chia chúng ra ngoài và đó là những gì mà điểm cuối của cụm / vùng chứa đó thực hiện. Đúng. Bây giờ chúng ta có hai cái đang chạy, vậy tại sao chúng ta không chạy một phần bổ sung đầy đủ, và để làm được điều đó, chúng ta sẽ xem xét một dịch vụ toàn cầu và tôi muốn sử dụng làm cơ sở để đưa vào mô-đun tiếp theo của chúng ta bởi vì chúng ta ' chúng ta sẽ bắt đầu nói về các chế độ hoặc loại dịch vụ, một trong số đó là dịch vụ toàn cầu, nhưng sau đó chúng ta còn có thêm hai loại nữa là các loại công việc, một công việc sao chép và một công việc toàn cầu.

Chạy công việc trên cụm

Giới thiệu Chế độ dịch vụ với cái nhìn nhanh về các dịch vụ toàn cầu

Vì vậy, bây giờ chúng tôi muốn chuyển sự chú ý của mình sang việc làm. Chúng tôi đã xem xét các dịch vụ hoạt động lâu dài và chúng tôi sẽ kết thúc ở đây và phân tích bằng cách xem xét một loại dịch vụ dài hạn khác và xem nó liên quan như thế nào đến các loại dịch vụ hoặc chế độ khác có liên quan đến công việc. Vì vậy, ngay bây giờ chúng tôi có hai phiên bản của dịch vụ echo của chúng tôi. Chúng tôi đã thấy cách DNS round robin có thể giúp chúng tôi tiếp cận với mọi trường hợp dịch vụ echo của chúng tôi. Chúng ta sẽ sử dụng kỹ thuật tương tự khi nói đến các công việc để triển khai một công việc thực hiện kiểm tra độ căng web của một dịch vụ và sử dụng tất cả các phiên bản của dịch vụ đó, vì vậy bạn muốn sử dụng cùng một phương pháp DNS round robin. Ngoài ra, bây giờ chúng tôi muốn xem cách tiếp cận vòng tròn này hoạt động như thế nào khi chúng tôi đưa nhiều nhiệm vụ trực tuyến hơn, cụ thể là chúng tôi muốn một nhiệm vụ trên mỗi node đơn lẻ vì sau đó chúng tôi sẽ có thể nhận được danh sách tất cả lịch sử vùng chứa của chúng tôi trên toàn bộ cụm với một chế độ xem web duy nhất. Đó là điều khó có được ngay bây giờ khi thiếu kết nối với từng node và thực hiện ps container **docker**. Vì vậy, chúng tôi sẽ thiết lập với một dịch vụ toàn cầu cho tiếng vang, và sau đó chúng tôi sẽ tiếp cận thực sự nhanh chóng và xem xét điểm cuối vùng chứa của cụm đó trông như thế nào. Và sau đó chúng ta sẽ sử dụng điều đó để xoay quanh việc nói về các công việc với một ví dụ về một công việc có chức năng tương tự. Vì vậy, tôi đã đi ra thư mục echo / stacks. Bên trong đây, chúng tôi có hai ngăn xếp đó, một ngăn xếp là toàn cầu và một là bản sao. Tôi sẽ loại bỏ nội dung của toàn cầu. Và thực sự, sự khác biệt chính ở đây là chúng ta chỉ cần đặt chế độ thành toàn cầu. Trên thực tế, tôi có thể chạy một điểm khác biệt ở đây và chúng ta có thể xem xét những điểm khác biệt đó. Và sự khác biệt chính là thay đổi chế độ. Vì vậy, tôi sẽ sao chép điều này bởi vì tôi muốn gắn với cùng một tên mà chúng tôi có ở đây để chúng tôi có thể sử dụng er mặc dù nó phải là ví dụ, chỉ để đơn giản hóa một số lệnh mà chúng tôi đã làm việc với; nếu không, tôi phải nói echo g\_echo và nó sẽ trở nên khó hiểu. Vì vậy, tệp toàn cầu đó chỉ để tham khảo. Hãy chỉnh sửa tệp bản sao của chúng tôi. Vì vậy, tôi đang ngồi ở đây bên trong Visual Studio Code, và chúng tôi sẽ chỉ thay thế mode: replicated và đặt nó thành mode: global, và sau đó tôi cần xóa dòng replicas vì điều đó không còn áp dụng nữa. Và sau đó ở đây tại dòng lệnh, tôi có thể xem nhanh các ngăn xếp của mình. Đó là ngăn xếp echor. Vì vậy, đó là những gì chúng tôi sẽ cập nhật. Bạn có muốn cho tôi biết chúng tôi làm điều đó như thế nào không? Chúng tôi đã không thực hiện nó trong một thời gian, nhưng chúng tôi chỉ cần thực hiện triển khai ngăn xếp **docker**, chuyển vào tệp bản sao của chúng tôi, tệp mà chúng tôi vừa chỉnh sửa và hãy giữ nguyên tên ở đây. Đây là điều tôi muốn chỉ ra và chúng ta sẽ giải quyết vấn đề theo cách khác, nhưng bạn không thể thay đổi chế độ dịch vụ sau khi đã triển khai một dịch vụ. Bạn nói cho tôi biết, bạn nghĩ chúng ta có thể làm gì? Chúng tôi chỉ có thể thực hiện một ngăn xếp **docker** rm, loại bỏ ngăn xếp echor của chúng tôi và bây giờ chúng tôi có thể tiếp tục và triển khai nó một lần nữa và đưa nó trở lại trực tuyến. Vì vậy, đó là giải pháp nhanh chóng của chúng tôi nếu chúng tôi không thể thay đổi một số giá trị mà chúng tôi muốn thay đổi.

Chuyển một node Drained sang Active Trigger Reconcration để lên lịch (các) nhiệm vụ cho (các) dịch vụ toàn cầu trên node Now Active!

Bây giờ, bạn có muốn đoán trình hiển thị trông như thế nào trước khi tôi chuyển sang nó không? Chúng tôi đây. Đây có phải là những gì bạn mong đợi? Vì vậy, có vẻ như chúng tôi có dịch vụ tiếng vọng trên các node khác nhau ở đây, ngoại trừ chúng tôi không có nó trên m2. Chúng tôi có thể làm mới ở đây và có vẻ như đó không phải là thứ chỉ đang cập nhật. Tại sao vậy? Chà, nếu bạn còn nhớ, chúng tôi đã làm một bản demo trước đó khi chúng tôi rút hết node m2, có nghĩa là nó sẽ không nhận được bất kỳ tác phẩm mới nào và đó là lý do tại sao nó không có phiên bản dịch vụ toàn cầu của chúng tôi. Bây giờ, điều thú vị về các dịch vụ toàn cầu, nếu tôi đưa tính khả dụng đó trở lại trực tuyến, để tôi có thể cập nhật node đó, tôi cần đặt tính khả dụng ở đây và chúng tôi sẽ đặt nó thành hoạt động và chúng tôi sẽ nhắm mục tiêu m2 ở đây, đảm bảo rằng bạn làm điều đó từ một node người quản lý. Bạn có thể thấy m2 hiện đang Hoạt động. Và bạn nghĩ chúng ta sẽ thấy gì ở đây khi chuyển sang trình duyệt? Nhìn kìa. Bây giờ chúng tôi đã có nó trên tất cả các node của chúng tôi. Vì vậy, nếu bạn muốn, bạn có thể để nó ráo nước hoặc bạn có thể đưa nó trở lại trực tuyến. Và bây giờ, cho đêm chung kết lớn, nếu tôi truy cập trình duyệt và làm mới ở đây, bạn nghĩ chúng ta sẽ thấy gì? Hãy cùng tìm hiểu tại đây. Vì vậy, chúng tôi đã có task.echor\_echo làm miền của chúng tôi. Nhìn kìa. Đây là lịch sử vùng chứa của chúng tôi và hy vọng không có gì làm tôi khó xử ở đây, trên toàn bộ cụm của chúng tôi. Điều đó khá gọn gàng để xem và một cái gì đó khá hữu ích, nếu bạn nghĩ về nó. Và một điều tôi chưa bao giờ thực sự chỉ ra, nhãn là thứ được sử dụng đằng sau hậu trường để giúp theo dõi đối tượng nào thuộc về một ngăn xếp, ngăn xếp cụ thể nào mà chúng thuộc về, ví dụ: ở đây là echor. Chúng tôi sẽ phóng to và chuyển sang một bên ở đây. Vì vậy, đây là một phần của cách chế độ bầy đàn **Docker** tìm ra đối tượng nào thuộc về ngăn xếp nào. Bằng cách đó, ví dụ: nếu bạn chỉnh sửa tệp ngăn xếp của mình và chỉ thay đổi tên của một dịch vụ, thực sự có một lệnh sơ lược có thể xuất hiện và xóa các phiên bản cũ của nó vì rất khó để biết bạn có thay đổi tên hay không. chính xác thì điều gì đang xảy ra vậy. Tệp ngăn xếp không còn tên cũ nữa, vậy làm cách nào để biết tên cũ là gì? Chà, chúng ta phải giữ nó ở đâu đó và chúng ta đã có nó trong các nhãn ở đây, vì vậy chúng ta có thể truy vấn tất cả các vùng chứa khác nhau đang chạy có một vùng tên ngăn xếp nhất định phù hợp với không gian tên của ngăn xếp của chúng ta, echor, và sau đó nếu chúng không còn được áp dụng nữa, chúng tôi chỉ có thể cắt bỏ chúng, tạo bất kỳ cái mới nào và tiếp tục ngày của chúng tôi. Vì vậy, đây là một phần của quá trình theo dõi trạng thái hiện tại là gì để chúng ta có thể điều chỉnh nó với các trạng thái mong muốn trong tương lai. Bây giờ, trước khi chúng ta kết thúc, tôi muốn quay lại phần đầu ở đây và tôi đã phóng to để bạn có thể thấy, chúng tôi có từng node. Tôi muốn đi qua càng nhanh. Chúng ta có m2, chúng ta có 2, m1, w1, vì vậy bạn lưu ý, chúng không theo thứ tự, w4, m3 và w3. Chúng không theo thứ tự. Đây có thể là thứ tự mà họ đã quay lại sau quá trình tra cứu DNS của chúng tôi. Và nếu bạn làm mới trang này, có thể bạn sẽ nhận được một đơn đặt hàng khác. Đây, m2, m2 là 22 đầu tiên, và tôi không nhớ những thứ đó theo thứ tự. Làm mới lại, bạn bắt đầu, m1 và w2, vì vậy mọi thứ đang lộn xộn xung quanh, giống như phản hồi vòng lặp gửi lại vùng chứa của chúng tôi khi nó yêu cầu tra cứu địa chỉ IP cho miền này để nó có thể đi sau đó liên hệ với những các volume khác nhau. Và nếu tôi mở một tab mới, dán vào phần tra cứu DNS cho tests.echor\_echo của chúng tôi, bạn nghĩ chúng ta sẽ quay lại đây khi tôi chạy cái này là gì? Hãy cùng tìm hiểu. Vì vậy, chúng tôi hiện đang sử dụng 10.0.3 làm mạng của mình và đó là vì chúng tôi đã xóa và tạo lại ngăn xếp của mình. Và điều thú vị là, chúng tôi nhận lại danh sách bảy địa chỉ IP khác nhau, một địa chỉ cho mỗi vùng chứa của chúng tôi. Nếu bạn vào đây và chỉ đặt echor\_echo, điều gì sẽ trở lại? Vì vậy, trong trường hợp này, chúng tôi chỉ nhận được một địa chỉ IP. Đây, một lần nữa, sẽ là vip của chúng tôi, là một phần trong cân bằng tải của chúng tôi trên mạng mặc định echor nội bộ của chúng tôi. Vì vậy, chỉ hữu ích khi xem điều này thay đổi như thế nào khi bạn mở rộng quy mô và thu nhỏ dịch vụ của mình. Đó là một cách thú vị để chơi và tìm hiểu một chút về cân bằng tải và thậm chí cả DNS round - robin thay vì cân bằng tải vip.

Tình huống: Sử dụng **Swarm** Jobs để kiểm tra căng thẳng Dịch vụ **Swarm** qua mạng lớp phủ phụ trợ

Bây giờ tôi muốn chuyển sự chú ý của chúng tôi sang công việc và vì vậy, chúng tôi sẽ có một thiết lập khác một chút vì chúng tôi sẽ chuyển trở lại bánh mâm xôi của chúng tôi. Tôi đã dành chút thời gian để dọn dẹp chúng một chút để thực sự tất cả những gì chúng tôi có ngay bây giờ là dịch vụ Viz đang hoạt động và tôi loại bỏ tất cả lịch sử mà tôi có thể tìm thấy trên bất cứ thứ gì có thể. Vì vậy, chúng tôi đang ở trong trạng thái sạch sẽ khá tốt như thể tôi vừa tạo lại cụm vì những gì tôi muốn làm là tiếp tục và triển khai một tập hợp các dịch vụ toàn cầu hoặc sao chép, và chúng tôi sẽ bắt đầu với quy mô nhỏ, chúng tôi sẽ không có tất cả năm để bắt đầu, nhưng cuối cùng chúng tôi sẽ làm. Tôi có ứng dụng kiểm tra mức độ căng thẳng này, đó là một ứng dụng node.js, đó là một image đa nền tảng , vì vậy bạn có thể tự do chạy nó trên máy ảo hoặc chơi với **Docker** hoặc thiết lập của riêng bạn theo cách nào bạn muốn, Tôi sẽ chạy nó trên bánh nướng và nó có một vài phương pháp, một trong số đó thực sự nhanh, không chiếm nhiều thời gian CPU về máy chủ và phương pháp còn lại có sự chậm trễ có chủ ý bên trong nó để làm chậm mọi thứ. Và như bạn có thể tưởng tượng, hai yêu cầu web càng nhanh có lẽ sẽ tạo ra thời gian bật tốt hơn và số liệu tốt hơn khi chúng tôi đặt nó ở chế độ tải. Nhân tiện, ứng dụng web kiểm tra căng thẳng này, tôi sẽ xuất bản một cổng để chúng ta có thể xem một số yêu cầu web, nhưng điều đó không cần thiết, đó là tùy chọn. Điều quan trọng là cổng 3000 là những gì chúng đang chạy và hiển thị với các mạng nội bộ cho các công việc mà chúng tôi khởi chạy. Bây giờ công việc có thể là bất cứ điều gì. Tôi chỉ nghĩ một ví dụ gọn gàng để thực hiện một kịch bản kiểm tra căng thẳng, ném nó vào một cụm nơi chúng tôi khởi chạy một loạt các dịch vụ hoặc một dịch vụ cùng một lúc, có thể là hai, sau đó ba và xem hiệu suất như thế nào, và vậy thì tại sao chúng ta không cũng trong suốt cụm đó, một số công việc hàng loạt để thực hiện một số thử nghiệm tải đối với các yêu cầu web đó. Và trong quá trình này, chúng ta sẽ kết thúc việc chia sẻ một mạng lớp phủ tùy chỉnh. Điều này sẽ được xác định bên trong dịch vụ của chúng tôi, vì vậy nó sẽ được gọi là sr\_testers, vì vậy đó là phía dịch vụ của phương trình và sau đó khi chúng tôi khởi chạy phía công việc của phương trình, chúng tôi sẽ sử dụng image Apache httpd với - trong lệnh kiểm tra tải ab, vì vậy đó sẽ là cấp công việc của chúng tôi, nếu bạn muốn, và sau đó chúng tôi sẽ kết nối ngẫu nhiên với cùng một mạng SR\_TESTERS và thực hiện các yêu cầu web vì chúng tôi sẽ thông qua trình cân bằng tải nội bộ đó nên nó không thực sự biết dịch vụ nào sẽ đáp ứng yêu cầu của nó. Và sau đó khi chúng tôi cảm thấy thoải mái với một công việc và có thể là một dịch vụ, chúng tôi sẽ bắt đầu mở rộng quy mô từng công việc. Ví dụ: chúng tôi cũng sẽ bổ sung thêm nhiều công việc và họ sẽ tiếp cận và kiểm tra các phiên bản khác của dịch vụ. Và nếu cân bằng tải của chúng tôi khá đồng đều, chúng tôi sẽ có thể đạt được tất cả các node khác nhau này phần nào đồng ý để nhận được một số kết quả thử nghiệm thú vị, hy vọng. Tôi hy vọng khi chúng tôi mở rộng quy mô, chúng tôi sẽ thấy một số cải tiến tuyến tính trong một số thời gian phản hồi, ít nhất là đối với thời gian bị ràng buộc bởi CPU vì nếu chúng tôi thêm nhiều vùng chứa hơn để xử lý các yêu cầu, chúng tôi sẽ có thể phần nào mở rộng quy mô công suất giống như một máy chủ web ngoài đời thực. Vì vậy, dù sao, đây là một trong những ý tưởng về những gì chúng ta có thể làm với công việc. Giới hạn của bầu trời. Chúng tôi có thể tiếp cận và kiểm tra máy chủ của người khác trên internet. Chúng tôi có thể thực hiện một số công việc hàng loạt, có lẽ là một số mô phỏng Monte Carlo cho các mục đích tài chính. Mọi thứ đều có thể. Chúng tôi có thể thực hiện bảo trì node. Vì vậy, chúng tôi có thể đi vào các node tròn của mình nếu chúng tôi cấp quyền truy cập vào các vùng chứa công việc này và để chúng dọn dẹp mọi thứ, chẳng hạn như có thể là các vùng chứa cũ hoặc image không được sử dụng hoặc các khối lượng đang treo lơ lửng. Vì vậy, hãy giữ một tâm trí cởi mở. Những gì tôi nghĩ ra cho một kịch bản công việc chỉ là một trong nhiều điều bạn có thể làm. Công việc chỉ là một khái niệm chạy đến hoàn thành trái ngược với một khái niệm dịch vụ chạy lâu dài .

Triển khai ngăn xếp sr của chúng tôi với một dịch vụ Đưa ứng dụng web vào kiểm tra căng thẳng với công việc

Đầu tiên, chúng tôi sẽ triển khai dịch vụ mà chúng tôi sẽ thử nghiệm. Và vì điều đó, tôi đang ngồi bên trong thư mục stress, nó nằm bên trong các ví dụ về ngăn xếp của chúng tôi, và bên trong đó tôi có một tệp stress - replica.yml. Vì vậy, đây là ngăn xếp của chúng tôi mà chúng tôi sẽ triển khai và tôi đã thiết lập tất cả nên chúng tôi không phải lo lắng quá nhiều về điều này. Nó sẽ sử dụng image này của tôi mà tôi đã xây dựng với một bộ thử nghiệm địa chỉ web bên trong nó. Hai yêu cầu web mà tôi đã thiết lập, một yêu cầu chậm và một yêu cầu nhanh. Bên trong nó lắng nghe trên cổng 3000 và chúng tôi sẽ ánh xạ đến cổng 3000 trên cụm để chúng tôi có quyền truy cập vào nó từ bất kỳ node nào nếu chúng tôi tò mò về các yêu cầu trông như thế nào. Trước tiên, chúng tôi đang triển khai dịch vụ này như một dịch vụ nhân bản và chúng tôi sẽ bắt đầu với một phiên bản để bắt đầu. Bây giờ đây là những gì mới. Chúng tôi đã liệt kê ra mạng lưới người thử nghiệm đặc biệt này mà tôi đang xác định và đặt tên rõ ràng và rõ ràng. Tôi có thể chọn mặc định đi kèm với dịch vụ cụ thể này, nhưng tôi muốn đặt cho nó một cái tên thông minh. Tôi cũng muốn cho bạn thấy bạn có thể tạo mạng của riêng mình. Bây giờ để làm điều đó, tôi cần đi xuống phần tử mạng ngang hàng với phần tử dịch vụ trong tệp ngăn xếp vì tệp ngăn xếp có thể xác định nhiều hơn chỉ các dịch vụ. Chúng ta có các mạng ngầm và chúng ta cũng có thể có các mạng rõ ràng. Vì vậy, tôi đã có mạng người thử nghiệm của mình và bằng cách này, nó sẽ có tiền tố là sr, đó là tên của ngăn xếp mà tôi muốn sử dụng và tôi rõ ràng đặt điều này là có thể đính kèm để tôi có thể thêm vào các vùng chứa sau này và làm nổi bật thực tế là không phải lúc nào mạng cũng có thể kết nối được. Ví dụ, mạng xâm nhập, bạn không thể gắn các vùng chứa độc lập vào nó. Trong trường hợp này, chúng tôi chỉ định điều này một cách rõ ràng, vì vậy chúng tôi sẽ có thể đính kèm bất cứ thứ gì chúng tôi muốn vào mạng này, bao gồm cả công việc của chúng tôi. Được rồi, vậy hãy triển khai cái này và tôi muốn gọi cái này là sr. Ngữ cảnh của tôi được đặt thành pi1 và chúng tôi sẽ chạy điều này và xem điều gì sẽ xảy ra. Được rồi, có vẻ như chúng tôi đã tạo sr\_testers mạng của mình và chúng tôi đã tạo dịch vụ sr\_web của mình. Vì vậy, nếu tôi liệt kê các mạng, bạn có thể thấy chúng tôi có mạng sr\_testers, nhưng chúng tôi không có mạng sr\_default và điều đó trái ngược với viz nơi chúng tôi có mạng viz\_default. Đây là mạng ngầm định, thay vào đó chúng ta có một mạng rõ ràng. Điều tôi muốn chỉ ra trong một tab mới ở đây, kiểm tra mạng **docker** trên viz\_default đó, hãy tìm từ Attachable. Bạn sẽ thấy mạng cụ thể này không thể truy cập được và đó là lý do tại sao tôi chỉ định rằng tôi muốn mạng của chúng tôi có thể truy cập được. Bây giờ nếu bạn kiểm tra mạng của chúng tôi, đó sẽ là sr\_testers, hãy tìm cùng một Attachable, bạn có thể thấy nó hiện được đặt thành true. Vì vậy, bạn có toàn quyền kiểm soát đối với những gì chính xác mạng được cấu hình. Bạn thậm chí có thể đặt những thứ như cấu hình IP hoặc bạn có thể thêm nhãn nếu muốn. Và nhân tiện, hãy nhìn xem, đây là một nhãn ở đây chỉ định không gian tên ngăn xếp là sr, vì vậy nó giúp nhóm điều hòa trạng thái hiện tại với trạng thái mong muốn, nếu chúng ta thay đổi mạng này. Nó cũng giúp dọn dẹp khi chúng tôi loại bỏ một ngăn xếp. Được rồi, đó là mạng của chúng tôi. Và nếu tôi thực hiện một ps ngăn xếp **docker** đơn giản ở đây, tôi có thể xem xét ngăn xếp sr của chúng tôi và bạn có thể thấy chúng tôi có một phiên bản được thiết lập và chạy như chúng tôi yêu cầu. Và nhân tiện, nếu bạn từng tò mò về các dịch vụ có sẵn cho ngăn xếp, tôi đã không xem xét điều này sớm hơn, nhưng thực sự có một lệnh dịch vụ ngăn xếp, như ps ngăn xếp, vì vậy ngăn xếp ps là dành cho các tác vụ, tốt, Điều gì sẽ xảy ra nếu bạn muốn biết những dịch vụ nào bên trong một ngăn xếp vì một ngăn xếp có thể có nhiều dịch vụ, hoặc có thể bạn chỉ muốn biết tên của nó là gì? Chà, bạn cũng có thể đặt sr ở cuối ở đây, và nó sẽ hiển thị cho bạn các dịch vụ của ngăn xếp này, trong trường hợp này là dịch vụ sr\_web của chúng tôi. Vì vậy, bạn sẽ có toàn bộ tên nếu bạn muốn truy cập trực tiếp sau đó. Và bạn có thể thấy trong trường hợp này, chúng tôi đã xuất bản cổng của mình, 3000, vì vậy chúng tôi có thể đưa ra yêu cầu đối với điều đó. Hãy mở một tab trong trình duyệt ở đây, và tôi sẽ đi đến pi1.lan, và 3000 là cổng. Và sau đó tôi cũng có một điểm cuối kiểm tra rất giống điểm cuối tiếng vọng của chúng tôi trong vùng chứa khác của chúng tôi. Nó chỉ in ra thông tin về người thực sự đang xử lý yêu cầu. Và nó cung cấp cho chúng tôi cùng một thông tin mà chúng tôi đã thấy trước đây, đó là địa chỉ cục bộ so với địa chỉ từ xa. Đây là VIP của chúng ta, đây là phiên bản container thực tế xử lý yêu cầu và đây là cổng. Vì vậy, mọi thứ có một chút thay đổi so với ví dụ khác, nhưng hầu hết là cùng một thông tin. Nhưng đó không thực sự là điểm cuối mà chúng tôi tò mò. Thay vào đó, một trong những người tôi muốn tiếp cận là khách hàng này - gật đầu và sau đó / 1. Vì vậy, đây là một trong những yêu cầu web không có độ trễ và nó chỉ in lại khách hàng giả mạo này. Và sau đó, yêu cầu khác mà tôi muốn thực hiện, nó sẽ không quá khác biệt ở đây, nó chỉ là / db\_block\_slow, để mô phỏng một kết nối cơ sở dữ liệu chậm để trả về cùng một thông tin. Và bạn thực sự sẽ không nhận thấy sự khác biệt vì chúng ta đang nói về độ trễ 10 mili giây ở đó , nhưng sẽ đủ khi chúng ta đến thời gian thử nghiệm để thấy sự khác biệt, đặc biệt là khi chúng ta bắt đầu thực hiện hàng trăm và hàng nghìn yêu cầu. .

Khám phá cách tạo công việc toàn cầu đầu tiên của chúng ta

Được rồi, bây giờ cho lần thử việc đầu tiên của chúng ta. Tôi đã thiết lập thư mục việc làm này, không phải là thư mục chuẩn bị việc làm, đó chỉ là một số ghi chú mà tôi có sẵn khi xem qua mọi thứ ở đây, tôi thậm chí có thể xóa nó trước khi khóa học này kết thúc, đó là một loạt các ví dụ mà tôi có thể muốn đi qua. Nhưng dù sao đi nữa, tôi sẽ sao chép mọi thứ vào thư mục việc làm này khi chúng tôi thử chúng và cái đầu tiên tôi muốn thử là 00 - abv.sh. Những gì điều này sẽ làm là chạy lệnh ab với cờ phiên bản, vì vậy abv, và tôi đã đánh số nó là 00, vì vậy chúng tôi sẽ có một chuỗi các công việc mà chúng tôi thực hiện và tôi đã có điều này bên trong một trình bao script, đơn giản vì nó sẽ là một lệnh mà chúng tôi chạy và tôi muốn cung cấp nó để bạn sao chép và dán. Tôi cũng muốn nói về một thực tế là có thể rất hấp dẫn khi cố gắng triển khai các công việc với một tệp ngăn xếp, và thật không may, điều đó là không thể vào lúc này. Điều đó nói rằng, có sự quan tâm đến tập hợp tính năng này, nơi bạn cũng có thể sử dụng tệp ngăn xếp. Thậm chí còn có một yêu cầu kéo đang chờ xử lý với chức năng, dự kiến sẵn sàng hoạt động và có một cuộc thảo luận đang diễn ra về điều này. Jobs vừa được bổ sung vào năm 20.10, tức là chỉ vào cuối năm 2020, vì vậy, một số tính năng vẫn chưa đạt về khả năng sử dụng, nhưng khung cơ bản đã có, và đó là một điều tốt vì đây là một tính năng đã được chờ đợi rất lâu. Như bạn có thể tưởng tượng, việc có các dịch vụ hoạt động lâu dài là quan trọng, và sau đó, việc có một số cơ chế để chạy các công việc thuộc nhiều loại khác nhau cũng quan trọng không kém. Vì vậy, hãy theo dõi và xem liệu hỗ trợ có sử dụng tệp ngăn xếp hay không và nếu có, bạn sẽ có thể điều chỉnh các ví dụ mà chúng tôi có ở đây để làm việc với tệp ngăn xếp vì bạn đã khá quen thuộc với tệp ngăn xếp, chúng tôi ' đã có rất nhiều ví dụ về những điều đó. Vấn đề chỉ là lấy các đối số mà chúng ta có trong tệp và đặt chúng vào đúng vị trí bên trong tệp ngăn xếp. Một vài điều về điều này. Tôi có một dịch vụ **docker** rm ở đây với abv, đó là tên của công việc mà tôi đang tạo. Tôi chỉ có cái này ở đây nếu bạn muốn chạy lại nhiều lần, bạn có thể xóa công việc ngay lập tức và tạo lại nó. Cũng làm nổi bật tầm quan trọng của tệp ngăn xếp so với cách tiếp cận theo tập lệnh. Tệp ngăn xếp cả hai phần chức năng này. Tôi coi đây là cách làm, và tệp ngăn xếp là nhiều hơn thế. Tôi muốn điều này xảy ra trên cụm của mình, tôi không quan tâm nó phải diễn ra như thế nào, nếu có một phiên bản của công việc, thì này, hãy loại bỏ nó. Tuy nhiên, chúng tôi chưa có chức năng đó, vì vậy chúng tôi cần phải đi sâu vào các API ở đây và nói, này, hãy xóa công việc nếu nó tồn tại, nếu không điều đó sẽ không gây ra bất kỳ vấn đề nào, nó sẽ chỉ gây ra lỗi nhanh chóng và sau đó chúng ta sẽ tiếp tục và tạo công việc lần đầu tiên, và lần sau khi bạn chạy công việc này, nó sẽ thực hiện công việc của mình và loại bỏ công việc cũ để tạo một công việc mới. Bây giờ, một điều quan trọng khác, như bạn có thể thấy, việc tạo công việc đòi hỏi phải sử dụng cùng một bộ lệnh như khi làm việc với các dịch vụ. Vì vậy, các công việc được lồng vào bên trong tập hợp các lệnh dịch vụ, và lúc đầu, tôi hơi bị loại bởi điều đó, nhưng thực tế là có rất nhiều điểm chung giữa cả hai, sẽ chỉ là rất nhiều sao chép / dán để thử và Hãy tách chúng ra và có một công việc **docker** chẳng hạn, và có thể điều đó sẽ tồn tại vào một ngày nào đó, nhưng nó thực sự sẽ không thay đổi cách bạn tương tác với nó. Bạn vẫn sẽ làm những việc như đặt tên cho công việc của mình, giống như đặt tên cho một dịch vụ. Bạn sẽ vẫn cần đặt loại công việc với đối số chế độ, vì vậy, công việc toàn cục trong trường hợp này. Hoặc chúng tôi đã sao chép các công việc và trong ví dụ này, tôi đang cho bạn thấy bạn vẫn có thể sử dụng tên máy chủ được tạo mẫu, sau đó tiếp theo chúng tôi cần cung cấp image và sau đó trong trường hợp này, chúng tôi có thể cung cấp một lệnh mà chúng tôi muốn chạy, ab và chúng tôi sẽ chỉ định - V cho phiên bản. Chỉ là một thử nghiệm rất đơn giản về việc chạy một lệnh sẽ in ra nhiều hơn hoặc ít hơn, chỉ cần trợ giúp cho lệnh ab và sau đó dừng lại. Điểm khởi đầu tốt đẹp để thiết lập và chạy thử nghiệm tải.

Thực hiện công việc đầu tiên của chúng tôi và tiến độ giám sát

Vì vậy, bây giờ, hãy tiếp tục và tạo công việc này theo cách bạn muốn. Tôi sẽ thực thi tập lệnh mà chúng tôi có. Nó sẽ xảy ra lỗi vì không có việc làm ngay bây giờ, và sau đó nó tạo ra năm trường hợp của công việc đó vì đó là một công việc toàn cầu. Họ đang chuẩn bị ngay bây giờ, vì vậy bạn sẽ nhận được thông tin trạng thái tốt khi các công việc này đang thực hiện. Nhân tiện, chuẩn bị có nghĩa là họ đang kéo image xuống. Vì vậy, thật tốt khi thực hiện một công việc toàn cầu trước tiên vì nó sẽ hâm nóng cụm của chúng tôi với image mà chúng tôi cần ở khắp mọi nơi. Vì vậy, khi chúng tôi muốn thực hiện kiểm tra hiệu suất, chúng tôi sẽ không giới thiệu biến liệu node có image hay không hoặc nó có phải kéo nó xuống không, liệu nó có thể tham gia với tất cả các node khác trong cùng một công việc không đã đến lúc thực sự căng thẳng để kiểm tra, hay chúng ta đã có một bài kiểm tra không đối xứng trong đó một trong các bài kiểm tra của chúng tôi chạy sau các bài kiểm tra khác và có thể một số đã chạy trước khi có lẽ chúng tôi cần tất cả chúng chạy gần nhất có thể cùng một lúc? Bằng cách này, chúng tôi biết rằng mọi người đều có image, đó là một rào cản đối với việc đồng bộ hóa các công việc, nếu điều đó quan trọng. Nó không phải lúc nào cũng quan trọng, nhưng trong trường hợp thử nghiệm có thể quan trọng. Được rồi, bạn có thể thấy một khi mọi thứ đã hoàn thành về tất cả các công việc của chúng tôi và bởi vì chúng tôi đã yêu cầu một công việc toàn cầu, việc hoàn thành sẽ được định nghĩa là cả năm công việc đều hoàn thành. Và thông báo rằng chúng tôi cũng hoàn thành công việc vào cuối. Và khi mỗi cái đang hoàn thành, bạn có thể thấy rằng chúng tôi đã có từ hoàn chỉnh hiển thị ở đây. Vì vậy, nó đi từ việc chuẩn bị, sau đó chạy và sau đó hoàn thành, và có thể cũng có một số trạng thái trung gian. Khi công việc đã hoàn thành, nó sẽ không chạy lại. Và khi chúng tôi kết thúc clip này, tôi vẫn đang suy nghĩ, bạn biết không, tôi đã đề cập rằng bảo trì node là một trường hợp sử dụng hợp lý cho một công việc. Chà, chỉ hâm nóng image thì sao? Bạn có cần một tập hợp các image được kéo xuống mỗi node không? Có lẽ đó là một nhiệm vụ tuyệt vời cho một công việc. Nó có thể đơn giản như triển khai image mà bạn cần, chạy lệnh trợ giúp, nhưng cũng có thể là triển khai một số loại lệnh sau đó thực thi kéo xuống một loạt image, có thể thông qua **Docker** Compose hoặc cách khác. Vì vậy, hey, một trường hợp sử dụng tuyệt vời khác.

Xem đầu ra công việc với **Docker** Service Logs JOBNAME

Chúng ta có thể làm gì để xem chính xác những gì đã xảy ra để xem liệu công việc của chúng ta có chạy thành công để thực hiện lệnh ab với cờ phiên bản hay chúng ta chỉ gọi một cái gì đó không tồn tại và gặp một số lỗi không mong muốn, và chúng ta không ' t thậm chí biết nó? Vì vậy, chúng ta có thể sử dụng lệnh dịch vụ **docker** để xem các nhật ký và sau đó tôi cung cấp những gì ở đây? Chà, chúng tôi đã gọi công việc của mình là abv, vì vậy chúng tôi chỉ cần cung cấp điều đó ngay tại đây và chúng tôi có thể nhận được tất cả nhật ký từ các trường hợp công việc của chúng tôi đã chạy. Và khi tôi nhìn vào kết quả ở đây, có vẻ như chúng ta đã có Đây là ApacheBench, Phiên bản 2.3. Vì vậy, có vẻ như chúng tôi đang khởi chạy đúng image với lệnh chính xác để truy cập công cụ kiểm tra A / B. Đó là tin tuyệt vời. Bây giờ, chúng sẽ được xen kẽ, không chắc chắn theo một thứ tự cụ thể. Ví dụ, đây là một số 2, đây là một số 5, đây là một vài trong số 1 và 3. Có vẻ như đây là những thứ chủ yếu theo thứ tự, nhưng bạn có nguy cơ bị xen kẽ. Vì vậy, một điều khác mà bạn có thể muốn làm, lệnh ghi nhật ký dịch vụ **docker** đó, để nhận được sự trợ giúp cho việc đó, bạn có thể thấy chúng tôi thực sự có thể cung cấp một dịch vụ hoặc một nhiệm vụ riêng lẻ. Vậy làm thế nào chúng ta có thể nhận được các nhiệm vụ cho công việc của mình?

Liệt kê các trường hợp công việc (còn gọi là Nhiệm vụ) với Dịch vụ **Docker** ps JOBNAME

Một lần nữa, chúng ta đang xử lý các giao diện dịch vụ này, vì vậy chúng ta chỉ cần xem qua lệnh ps tương tự để nhận các nhiệm vụ cho một công việc. Vì vậy, chúng tôi cần cung cấp công việc của mình. Hãy nhớ đó là gì? Đó là abv. Cứ như vậy, chúng tôi có thể lấy lại thông tin của mình về các nhiệm vụ khác nhau đã chạy hoặc ít nhiều các trường hợp công việc của chúng tôi. Chúng ta có thể lấy các mã nhận dạng ở đây, sau đó đi xuống bên dưới, và đó là những gì chúng ta có thể chuyển đến lệnh nhật ký nếu chúng ta muốn xem nhật ký cho một tác vụ duy nhất. Bây giờ một điều thú vị cần làm, ngoài việc xem xét từng nhật ký, chúng ta có thể tránh được vấn đề xen kẽ nếu chúng ta liệt kê thông qua các nhiệm vụ của mình và lần lượt gọi lệnh nhật ký. Và chúng tôi có một lệnh soạn sẵn ở đây để tăng tốc độ này. Điều này sẽ hoạt động trong Z shell bằng cách sử dụng một cú pháp nhỏ gọn, nhưng nó ít nhiều là vòng lặp for bắt biến t này từ đầu ra của lệnh cụ thể này, là một dịch vụ **docker** ps giống như chúng ta đã chạy trước đây với abv để chúng ta nhận được trở lại danh sách các nhiệm vụ cho công việc của chúng tôi. Và sau đó tôi thêm cờ - q. Bạn có nhớ điều đó làm gì không? Chà, điều đó chỉ giới hạn đầu ra cho số nhận dạng cho bất kỳ thứ gì mà chúng tôi đang liệt kê, trong trường hợp này, nhiệm vụ của chúng tôi bởi vì chúng tôi cần số nhận dạng đó để cung cấp, sau đó, cho lệnh ghi nhật ký dịch vụ **docker** của chúng tôi. Và bạn có thể thấy tôi đang làm điều đó ngay tại đây bằng cách đưa vào giá trị của biến t. Bây giờ, tôi đã thêm một tiếng vọng, vì vậy bạn có thể thấy chính xác điều này sẽ chạy trước khi tôi chạy nó. Như bạn có thể thấy, bây giờ chúng ta có năm lệnh gọi đến các nhật ký **docker**, mỗi lệnh có một phiên bản nhiệm vụ riêng biệt. Vì vậy, bây giờ mọi thứ sẽ phần nào được đặt hàng. Nếu tôi loại bỏ tiếng vang, tôi có thể chạy nó ngay bây giờ. Và bùng nổ, bùng nổ, bùng nổ, bùng nổ, bùng nổ, bùng nổ, bùng nổ, từng cái một, chúng tôi nhận được nhật ký cho mỗi nhiệm vụ. Và giờ đây vấn đề xen kẽ không còn là vấn đề đáng lo ngại nữa. Trên thực tế, bạn có thể bắt đầu thấy một số lặp lại ở đây, Đây là ApacheBench, Đây là ApacheBench. Chúng tôi đã có Bản quyền trên cả hai điều này, đã được Cấp phép, được Cấp phép. Cuộn lên đây, cùng một mẫu. Bản quyền này được cấp phép, bản quyền này được cấp phép, bản quyền này đã được cấp phép. Vì vậy, điều này sẽ dễ dàng hơn rất nhiều và tôi sẽ thu nhỏ nó lại một chút để nó không bị quấn. Điều này, bạn có thể thấy mô hình bây giờ, dễ hiểu hơn rất nhiều rằng chúng tôi có kết quả phù hợp đến từ tất cả các nhiệm vụ của chúng tôi và chúng tôi không gặp lỗi nào trong số đó, điều quan trọng vì có thể một trong số chúng có thể đã thất bại . Vì vậy, hãy luôn cảm thấy thoải mái khi sử dụng các thủ thuật lên tay áo của bạn. Ví dụ: nếu bạn quen thuộc với ngôn ngữ shell, bạn muốn viết một vòng lặp for. Đó là một cách tuyệt vời để tận dụng một lệnh yêu cầu ID tác vụ một lần để sau đó nhanh chóng sử dụng nó để khai thác nó nhằm tạo ra một đầu ra nhật ký sạch đẹp. Vì vậy, điều này sẽ trái ngược với nhật ký dịch vụ **docker** của chúng tôi, chuyển vào abv, về mặt kỹ thuật cũng làm điều tương tự. Nó chỉ không đảm bảo cho chúng tôi cùng một thứ tự. Ví dụ, bạn có thể thấy ở đây, chúng tôi có một số xen kẽ đang diễn ra giữa 5 và 2. Và 5, nhiệm vụ là hiển thị node nào nó đã chạy trên. Hãy nhớ rằng, đây là một công việc toàn cầu. Vì vậy, một sẽ chạy trên mỗi node. Và vì vậy, mặc dù ở đây có thể gây hiểu nhầm rằng 3 và 4 trông giống như chúng theo đúng thứ tự và 1 cũng vậy, có vẻ như 2 và 5 đã bắt đầu gần như đồng thời và thay đổi nhau về sản lượng ở đây. Ở trên này, chúng tôi không có vấn đề đó.

Phiên bản công việc đầu tiên giao tiếp qua mạng lớp phủ do người dùng xác định để tải kiểm tra dịch vụ của chúng tôi!

Vì vậy, bây giờ chúng tôi sẽ chạy qua thử nghiệm tải thực tế đầu tiên của chúng tôi tại đây đối với dịch vụ sr\_web của chúng tôi. Vì vậy, một phiên bản đang chạy bây giờ xảy ra trên pi2. Nó không thực sự quan trọng. Chúng tôi sẽ khớp điều đó với một công việc và nó sẽ gọi ra và thực hiện một bài kiểm tra rất cơ bản chỉ để đảm bảo rằng chúng tôi có thể kiểm tra chức năng và quan trọng hơn, giao tiếp với dịch vụ sr\_web của chúng tôi. Được rồi, cho bài kiểm tra, tôi đang ngồi đây bên trong thư mục việc làm của mình và tôi đã sao chép một trong các ví dụ khác. Chúng tôi có 01 - ab1.sh. Điều đó là viết tắt của việc chạy thử nghiệm A / B trên một bản sao. Hãy xem kịch bản đó. Cũng giống như trước đây, tôi có quyền xóa, vì vậy bạn có thể chạy đi chạy lại điều này. Cái này được đặt tên là ab1. Một sự khác biệt lớn là đây là một công việc được nhân rộng, không phải là một công việc toàn cầu, bởi vì chúng tôi chỉ muốn một ví dụ. Chúng tôi không muốn năm. Tuy nhiên, tôi chỉ định điều đó một cách rõ ràng bên dưới. Đó là tùy chọn vì nó sẽ là 1 theo mặc định. Tham gia lại mạng sr\_testers, giống như trước đây, image httpd. Cũng vậy thôi. Ab là lệnh tương tự. Tuy nhiên, bây giờ chúng ta có những lập luận mới. Chúng tôi có - n cho số lượng yêu cầu mà chúng tôi sẽ gửi, - c cho đồng thời. Vì vậy, chúng tôi sẽ gửi 100 yêu cầu với tối đa 10 yêu cầu cùng một lúc và đây là điểm cuối mà chúng tôi sẽ đạt được. Lưu ý, chúng tôi đang tìm kiếm trên sr\_web. Điều này sẽ trông rất quen thuộc. Chúng ta hãy đi vào phần dài để nói về cách các dịch vụ của chúng tôi có thể nói chuyện với nhau hoặc trong trường hợp này, một công việc và dịch vụ của chúng tôi có thể nói chuyện với nhau thông qua DNS. Điều này sẽ được dịch sang cái gì? Chà, nó sẽ được dịch sang bộ cân bằng tải chạy trên mạng nội bộ của chúng tôi. Sau đó, nó sẽ cung cấp cho chúng tôi một IP chỉ được phân phối ngẫu nhiên đến một trong các vùng chứa của chúng tôi. Trong trường hợp này, chúng ta chỉ có một con chạy, vì vậy nó sẽ chuyển đến cái trên pi2. Tất nhiên, cổng 3000, và đây là điểm cuối của gật đầu. Vì vậy, quá trình này sẽ không mất nhiều thời gian và chúng ta sẽ có thể thấy một số đầu ra khi chúng ta hoàn thành. Vì vậy, tôi sẽ xóa màn hình ở đây và tôi sẽ chạy nó. Nó bị lỗi bởi vì chúng tôi chưa có nó. Tốt rồi. Và bây giờ chúng tôi đang thực hiện nhiệm vụ của mình. Nó đang bắt đầu và nó đang lặp đi lặp lại, hy vọng đạt được điểm cuối đó và nắm bắt một số chỉ số cho chúng tôi. Được rồi, hãy để ý một số điều ở đây. Chúng tôi đã có tiến độ công việc, 1 trong số 1 đã hoàn thành. Về các nhiệm vụ hoạt động, trong một thời gian ở đó, có 1, và bây giờ không có. Và bây giờ chúng ta đã hoàn thành 1/1 và công việc của chúng ta được đánh dấu là đã hoàn thành. Bạn cũng có thể lưu ý, nếu bạn xem qua dịch vụ, vì vậy tôi sẽ liệt kê các dịch vụ của tôi ở đây, bạn có thể thấy chúng tôi có ab1 của chúng tôi. Và qua đây, 1/1 là hoàn thành. Nếu bạn nhìn xuống bên dưới trên abv, chúng tôi đã hoàn thành được 5/5. Và để có biện pháp tốt, hãy đảm bảo rằng thử nghiệm của chúng tôi thực sự hoạt động. Làm thế nào chúng ta có thể làm điều đó? Chà, nhật ký dịch vụ **docker** cũ tốt và ab1, trong trường hợp này. Chúng tôi chỉ có một chiếc đang chạy, vì vậy điều này không quá tệ để xem xét. Và thì đấy, chúng tôi nhận được đầu ra của mình, và đây là những gì chúng ta nên thấy ở đây. Đây là thông tin thời gian đến từ lệnh ab. Một lần nữa, đây là ApacheBench. Có vẻ như chúng tôi đã thất bại. Và nó đưa ra một số thông tin như thời gian thực hiện cho các bài kiểm tra, bao nhiêu hoàn thành, bao nhiêu thất bại, bao nhiêu yêu cầu mỗi giây. Nó thậm chí còn cung cấp cho chúng ta một bảng nhỏ về thời gian kết nối, xử lý, chờ đợi và tổng thời gian tối thiểu, trung bình, trung bình và tối đa. Vì vậy, đây là thử nghiệm tải của chúng tôi. Ở bên dưới, có lẽ là phần thú vị nhất khi chúng tôi muốn bắt đầu so sánh một số trong số này, tuy nhiên, đó thực sự không phải là điểm của khóa học này, nhưng bạn có thể thấy chúng tôi có thông tin thời gian cho các phân vị. Vì vậy, phân vị thứ 50 đạt tốc độ 22 mili giây, trong khi phân vị thứ 99, 56 mili giây. Vì vậy, tất cả, rất lớn về thời gian thử nghiệm.

Xác thực công việc của chúng tôi đang trao đổi với dịch vụ của chúng tôi qua Mạng lớp phủ sr\_testers

Trong ví dụ tiếp theo này, hãy giả sử rằng vì lý do nào đó mà chúng tôi không tin rằng các yêu cầu web của chúng tôi đang thực sự được thực hiện. Và hãy cố gắng làm điều gì đó để xác thực rằng chúng tôi có thể gửi một công việc thực sự có thể kết nối với dịch vụ sr\_web của chúng tôi và nhận lại phản hồi. Đối với ví dụ này, tôi có một hướng dẫn 02 - testy. Nó thực sự là một chuỗi các lệnh. Bạn không muốn loại bỏ tất cả điều này cùng một lúc. Điều đầu tiên nó yêu cầu bạn làm là những gì tôi đã làm ở đây ở phía bên phải. Tôi đã thu nhỏ một chút và tôi đã có lệnh này sẵn sàng hoạt động, kiểm tra nhật ký dịch vụ **docker**, và sau đó làm theo . Tôi sẽ tạo một công việc thử nghiệm và theo dõi nhật ký trên đó. Và tôi sẽ thực hiện một loạt lệnh thông qua nó, và tôi sẽ cho bạn thấy điều đó có thể xảy ra như thế nào, nhưng chúng ta cần bắn ra một lệnh để chúng ta có thể tiếp tục và làm theo mọi thứ ở đây. Nếu tôi chạy điều này ngay bây giờ, bạn có thể thấy chúng tôi chưa có dịch vụ hoặc tác vụ hoặc công việc, trong trường hợp của chúng tôi, vì vậy chúng tôi cần đợi công việc được tạo. Tôi đã chạy qua một số kịch bản thử nghiệm, vì vậy image sẽ được sử dụng cho việc này đã được làm ấm trên cụm pi nhỏ của tôi. Bạn sẽ mất một chút thời gian khi lần đầu tiên bắt đầu chạy phần mềm này và công việc được lên lịch trên các node khác nhau. Đừng lo lắng nếu mất một chút thời gian để kéo mọi thứ xuống trạng thái chuẩn bị. Nó thực sự sẽ cung cấp cho bạn nhiều thời gian để bắt đầu theo dõi trên nhật ký ở phía bên tay phải ở đây. Vì vậy, hướng dẫn này, nếu tôi đưa nó ra đây, có một loạt lệnh. Đây là nơi bạn có thể lấy lệnh theo dõi nhật ký , và đây là lệnh tiếp theo chúng ta sẽ chạy. Và tôi thực sự sẽ đặt nó ở phía bên tay phải ở đây trong Visual Studio Code để tôi có thể sao chép mọi thứ. Điều đầu tiên chúng tôi muốn chạy là tạo công việc khó khăn của chúng tôi. Nó sẽ là một công việc được nhân rộng. Sẽ chỉ có một trường hợp của nó. Nó sẽ kết nối với mạng sr\_testers của chúng tôi, vì vậy mô phỏng mọi thứ giống như các công việc chúng tôi đã chạy cho đến nay. Và sau đó sẽ tình cờ sử dụng image giống như dịch vụ trang web đang sử dụng, và đó là bởi vì loại image con dao của Quân đội Thụy Sĩ. Tôi có một loạt các tiện ích được cài đặt nhỏ như đào, và cuộn, và các tiện ích IP khác nhau sẽ giúp bạn khắc phục sự cố hoặc tìm hiểu kỹ mọi thứ. Vì vậy, nó có nhiều mục đích ngoài việc chỉ chạy máy chủ web đó. Nó cũng có thể chạy một loạt các công cụ dòng lệnh khác nhau. cụ thể, những gì chúng tôi sẽ làm đầu tiên là chạy lệnh đào và chuyển dọc theo sr\_web dưới dạng miền mà chúng tôi sẽ tìm kiếm, giống như chúng tôi đã chuyển nó vào ứng dụng web của mình trong mô-đun cuối cùng và nhận được phản hồi trở lại trong trình duyệt với địa chỉ IP. Được rồi, vì vậy tôi sẽ loại bỏ điều này và nhanh nhất có thể, hãy đến đây và bắt đầu các nhật ký tiếp theo, và có vẻ như tôi đã hoàn thành. Công việc ở đây đang bắt đầu, và này, này, chúng tôi đã có một số đầu ra ở đây. Và bạn có nhìn vào nó không? Chúng tôi đã có 10.0.2.2. Điều đó đã có vẻ tốt đối với tôi về mặt hiểu biết, đây chắc chắn không phải là mạng xâm nhập. Và tôi có thể đào sâu vào 0,2 này và tìm ra đó là gì. Tôi cho rằng chúng ta có thể đến đây, giải quyết vấn đề này, liệt kê các mạng của chúng ta và xem mạng sr\_testers của chúng ta có gì. Và nó thực sự có 10.0.2.1 của chúng tôi, vì vậy đó là mạng cho sr\_testers của chúng tôi. Vì vậy, điều đó đã được xác thực, mảnh đầu tiên của câu đố có thể giao tiếp qua mạng sr\_testers như tôi nghĩ. Phần tiếp theo của câu đố, tại sao chúng ta không có một chút thú vị ở đây? Tại sao chúng ta không thấy nó như thế nào khi mở rộng quy mô lên ba trường hợp công việc của chúng ta? Bạn nghĩ điều gì sẽ xảy ra ở phía bên tay phải ở đây? Và tôi sẽ làm rõ điều này để tránh bất kỳ sự nhầm lẫn nào. Vì vậy, chúng tôi sẽ mở rộng quy mô công việc khó khăn của mình lên đến 3 trường hợp. Hãy cùng tìm hiểu. Vì vậy, nó thực sự bắt đầu ba công việc riêng biệt. Khi nó hoàn tất, phần tiếp theo sẽ được bắt đầu. Và nếu bạn thấy ở đây trong đầu ra, chúng tôi có một đơn vị phản hồi, hai đơn vị, chúng tôi sẽ sớm có một phần ba ở đây và chúng tôi có đơn vị thứ ba. Và có vẻ như cả ba công việc của chúng tôi đã hoàn thành. Vì vậy, khi chúng tôi mở rộng quy mô lên ba phiên bản, nó thực hiện cùng một công việc mà chúng tôi đã thực hiện và thực hiện nó ba lần.

Thu nhỏ công việc và xác thực Chúng tôi có thể thực hiện công việc có thể gửi yêu cầu web thành công

Được rồi, tiếp tục. Tôi sẽ giải quyết cả hai bên ở đây. Chúng tôi vẫn đang theo dõi ở phía bên phải , đó là một điều thuận tiện khi bạn tiếp tục theo dõi. Hãy tiếp tục và hãy thu nhỏ quy mô, và bạn nghĩ điều gì sẽ xảy ra khi tôi làm điều này? Vì vậy, chúng ta sẽ từ 3 xuống 1. Chà, điều sắp xảy ra là chúng ta sẽ thực hiện một ví dụ về công việc của mình. Vì vậy, bất cứ khi nào bạn cập nhật công việc của mình, bạn sẽ kích hoạt nó chạy lại. Được rồi, chúng ta hãy làm rõ điều này một lần nữa, xem hướng dẫn của tôi, và chúng ta có gì ở đây tiếp theo? Chà, chúng tôi có một trò vui. Đây là bản cập nhật sẽ thay đổi args, sẽ thay đổi lệnh đang chạy. Vì vậy, thay vì lệnh đầu tiên mà chúng ta có, đó là dig sr\_web, lệnh này sẽ chạy bash, thực sự truyền một lệnh con gồm hai lệnh. Đó là lý do tại sao tôi cần Bash ở đây. Bạn có thể thấy dấu chấm phẩy ở giữa hai lệnh. Lệnh đầu tiên là một cuộc gọi curl tới sr\_web của chúng tôi trên cổng 3000 và sau đó kiểm tra điểm cuối đó vì tôi muốn xem liệu chúng tôi có thực sự lấy lại một số dữ liệu từ trang web trông quen thuộc của chúng tôi hay không. Đó là phần cuối cùng của câu đố kiểm tra để đảm bảo rằng chúng ta đang thực sự giao tiếp. Và ngay sau đó tôi chỉ tò mò chạy một ip để hiển thị địa chỉ IP của mình. Vì vậy, tôi sẽ sao chép nó, quay lại đây, dán cái này ở phía bên trái và hy vọng tôi đã hiểu đúng như vậy, chạy cái đó, và chỉ cần đợi một giây ở đây. Và nhìn vào đó, chúng tôi đã có đầu ra của chúng tôi ở phía bên tay phải. Vì vậy, chúng ta hãy xem điều này bị phá vỡ ở đâu. Đây là đầu ra của lệnh gọi curl, và sau đó mọi thứ không có màu hồng là đầu ra của ip a s. Vì vậy, bạn thực sự có thể thấy ở đây địa chỉ IP của chúng tôi là 10.0.2.116. Chúng tôi sẽ đảo ngược điều này ngay bây giờ. Đây là tất cả các giao diện của chúng tôi ở dưới đây. Và ở trên nếu chúng ta nhìn vào đầu ra của lệnh gọi curl, đây là điểm cuối kiểm tra và bạn có thể thấy người trả lời đang ở trên 10.0.2.3 và yêu cầu đang chuyển qua 10.0.2.107. Vì vậy, bạn có thể tìm hiểu thông tin mạng và xem điều đó có nghĩa là gì, nhưng vào cuối ngày, điều quan trọng là chúng tôi sẽ lấy lại dữ liệu từ một trong các yêu cầu web của chúng tôi. Điều đó khiến tôi cảm thấy tự tin hơn rất nhiều rằng khi chúng tôi thực hiện kiểm tra tải, chúng tôi thực sự đang kiểm tra tải chứ không chỉ lấy lại trang lỗi đó. Hãy làm rõ điều này một lần nữa và hãy làm một điều nữa. Hãy tải xuống bản cập nhật cuối cùng này tại đây. Lần này chúng tôi sẽ thực hiện curl và chúng tôi sẽ đạt được một trong những điểm cuối mà chúng tôi đã thử nghiệm, nhanh hơn trong hai điểm. Vì vậy, tôi sẽ dán nó vào đây và xem đầu ra ở phía bên tay phải. Và này, hãy nhìn vào điều đó, nó hơi khó hiểu với các chỉ số tiến độ cuộn, nhưng chúng tôi có nội dung ở đây sẽ quay lại khi chúng tôi đưa ra yêu cầu này trong trình duyệt web. Vì vậy, bạn có thể thấy cùng một yêu cầu bên trong trình duyệt web bên ngoài đến cụm đến thông qua cổng đã xuất bản 3000. Và chúng tôi có cùng một cấu trúc JSON nhỏ, id 1, tên Bob, họ Doe, giống như ở đây tại dòng lệnh, id 1, tên Bob, họ Doe. Vì vậy, tôi khá tự tin rằng chúng tôi đang đạt được các điểm cuối mà chúng tôi mong muốn về kiểm tra tải. Vì vậy, chúng ta hãy tiếp tục và chúng ta hãy dọn dẹp ở đây. Làm thế nào để làm điều đó? Vâng, tôi chỉ có thể làm một dịch vụ **docker** rm testy. Ở phía bên tay phải , bạn sẽ chỉ cần Ctrl + C để thoát khỏi điều đó. Và bây giờ chúng tôi cảm thấy tự tin rằng chúng tôi đang thực sự giao tiếp với dịch vụ mà chúng tôi mong muốn qua mạng mà chúng tôi dự định.

Hình dung sự tương tác trong thử nghiệm tải tiếp theo của chúng tôi giữa **Swarm** Jobs và **Swarm** Services

Bây giờ chúng tôi đã xác nhận rằng các bài kiểm tra của chúng tôi là chính xác, chúng ta hãy bắt đầu xem mọi thứ mở rộng như thế nào. Và tôi đã quyết định một thiết kế mà chúng tôi có một công việc, và ban đầu chúng tôi có một công việc đó chỉ tương tác với một phiên bản dịch vụ của chúng tôi. Chúng ta sẽ xem xét điểm cuối chậm trên ứng dụng web đó và chỉ xem hiệu suất như thế nào khi ném 1000 yêu cầu vào nó, cho phép tối đa 100 yêu cầu cùng một lúc. Sau đó, chúng ta sẽ so sánh cùng một kịch bản đó với việc có hai vùng chứa dịch vụ. Chúng tôi vẫn sẽ có một công việc và sau đó chúng tôi sẽ thấy điều gì đó theo thứ tự cải thiện 50% hoặc giảm một nửa thời gian cần thiết để đưa ra yêu cầu. Và sau đó, chúng tôi sẽ tải lên năm container cho dịch vụ của mình và chúng tôi sẽ thấy lượng thời gian cần thiết để thực hiện các yêu cầu đó giảm đi năm lần.

Sử dụng **Swarm** Jobs để kiểm tra dịch vụ **Swarm** để xác minh giả thuyết về khả năng mở rộng

Bây giờ, để thực hiện bài kiểm tra này, tôi có một tệp hướng dẫn khác. Nó không có nghĩa là được thực hiện trực tiếp, nhưng nó có một loạt các bước trong đó mà bạn có thể làm theo. Đây là 03 - làm chậm - hướng dẫn. Tôi sẽ đóng cửa sổ này vì tôi muốn chia đôi màn hình của mình và tôi khuyên bạn nên làm như vậy nếu bạn đang theo dõi. Ở phía bên phải , chúng ta sẽ thực hiện lại nhật ký sau. Và hãy để tôi xem qua VS Code và lấy dòng ở đây. Đây là hướng dẫn 03 - chậm - . Tôi có một dòng ở đây tôi muốn bạn sao chép cho công việc mà chúng tôi sẽ sử dụng để thực hiện thử nghiệm của mình. Và cụ thể, đây là lệnh theo dõi nhật ký đầu ra thử nghiệm của chúng tôi ở phía bên tay phải tại đây. Sử dụng cùng một công việc trong suốt. Nó chỉ là một công việc và chúng tôi đang xem xét hiệu suất của một công việc đó so với 1, 2 và sau đó là 5 máy chủ, nhiều hơn hoặc ít hơn. Vì vậy, tất nhiên, chúng tôi chưa thể theo dõi điều này. Nó không tồn tại, vì vậy chúng ta cần phải chờ đợi. Ở phía bên tay trái , hãy đảm bảo rằng bạn chia tỷ lệ sr\_web xuống một bản sao. Trong trường hợp bạn đã mày mò với bất kỳ kích thước nào khác nhau, hãy đảm bảo rằng chúng tôi đang bắt đầu với phiên bản dịch vụ duy nhất mà chúng tôi muốn bắt đầu. Và tôi sẽ liệt kê các dịch vụ của mình. Có khá nhiều ở đây. Một điều chúng ta có thể làm, chúng ta có thể loại bỏ ab1 và abv. Chúng tôi đã hoàn thành những công việc đó. Điều đó sẽ giúp chúng tôi dọn dẹp nhà cửa một chút ở đây. Bây giờ khi chúng tôi liệt kê các dịch vụ của mình, chúng tôi chỉ có hai, sr\_web và viz\_web. Và chỉ cần xác nhận rằng bạn có 1/1 trên sr\_web. Và sau đó, hãy quay lại hướng dẫn của chúng tôi. Và bên trong đây, chúng ta nên có lệnh để khởi động công việc duy nhất của mình. Vì vậy, sao chép nó ra, và hãy dán nó vào đây, và chúng ta hãy nói chuyện này. Vì vậy, dịch vụ **docker** đơn giản tạo tên là onejob. Đó là một công việc được nhân rộng chỉ với một bản sao. Chỉ làm cho điều đó rõ ràng ở đây. Và sau đó chúng ta sẽ tham gia vào cùng một mạng sr\_testers bằng cách sử dụng lại image Apache, ab. Lần này 1.000 yêu cầu lên đến 100 yêu cầu cùng một lúc và chúng tôi sẽ sử dụng dịch vụ sr\_web trên cổng 3000 giống như trước đây, nhưng bây giờ chúng tôi sẽ truy cập vào db\_block\_slow này. Cái này có độ trễ 10 mili giây bên trong nó như thể chúng ta phải tính toán CPU khá chuyên sâu để thực hiện. Chúng ta sẽ thấy một số cải tiến khi chúng ta thêm nhiều vùng chứa dịch vụ hơn. Vì vậy, tôi sẽ loại bỏ điều này và đến ngay đây và theo dõi nhật ký của tôi, và đừng lo lắng nếu bạn không theo dõi ngay lập tức. Bạn sẽ có thể bắt kịp với bất cứ thứ gì đã được in ra. Công việc của chúng tôi đang bắt đầu lại ở phía bên trái của chúng tôi , và nó sẽ thực hiện các bài kiểm tra điểm chuẩn, và bạn có thể thấy rằng đầu ra đã được chuyển qua ở bên phải. Nó hơi nhỏ một chút, nhưng chúng tôi đã hoàn thành 600 yêu cầu cho đến nay và chúng tôi có tổng số 1.000, vì vậy chúng tôi sắp hoàn thành ở đây. Được rồi, này, có đầu ra của chúng tôi. Phân vị thứ 50 là khoảng 1300 mili giây và phân vị thứ 100 là 1593 mili giây. Và tôi sẽ đưa ra dự đoán ở phân vị thứ 50 cho thử nghiệm tiếp theo của chúng tôi. Nếu chúng tôi tăng gấp đôi dung lượng của mình, chúng tôi sẽ có thể giảm thời lượng xuống một nửa hoặc khoảng 670. Và khi chúng tôi tăng lên 5, hãy chỉ lấy 1335 trên 5. Khoảng 270. Bây giờ, trước khi bạn chạy thử nghiệm tiếp theo, tôi ' d khuyên bạn nên chuyển sang phía bên tay phải và nhấn phím Return một vài lần để thêm một số khoảng trống giữa đầu ra của thử nghiệm đầu tiên và các thử nghiệm tiếp theo. Nếu bạn xem qua tệp hướng dẫn của chúng tôi, chúng tôi có thể cuộn xuống đây và chúng tôi sẽ có thể cập nhật dịch vụ sr\_web của mình. Và lần này, chúng tôi sẽ cung cấp cho nó hai bản sao. Tôi cũng chỉ định ‑‑ bản sao - tối đa - trên mỗi node của 1 để đảm bảo rằng các node này được trải rộng trên các node riêng biệt để bạn nhận được lợi ích của Raspberry Pis riêng biệt, có mã lực riêng biệt đằng sau chúng. Điều này sẽ đặc biệt quan trọng nếu bạn có một số tác vụ thực sự chuyên sâu về tính toán và bạn cần đảm bảo rằng bạn có thêm dung lượng CPU. Bạn không muốn có hai phiên bản dịch vụ trên một node. Vì vậy, tôi sẽ chạy nó và chúng ta sẽ có hai phiên bản ở đây sau một giây. Tôi sẽ xác nhận điều đó với giao diện người dùng và xem xét điều đó. Chúng tôi có sr\_web và sr\_web ở đây ở phía bên tay phải trên hai trong số các node của chúng tôi, hai node công nhân của chúng tôi. Vậy thì tôi sẽ giảm thiểu điều này. Vì vậy, làm thế nào để tôi chạy lại thử nghiệm của chúng tôi khi biết rằng tôi muốn thực hiện các thử nghiệm giống hệt nhau ngay bây giờ, chỉ với hai máy chủ riêng biệt? Chúng tôi chỉ cần buộc cập nhật công việc của mình, vì vậy chúng tôi có thể thực hiện cập nhật dịch vụ **docker**, đặt cờ ‑‑ force vào và sau đó onejob là tên công việc của chúng tôi. Nếu bạn muốn sao chép, điều đó đã kết thúc trong tệp hướng dẫn, chính xác lệnh mà tôi vừa nhập. Vì vậy, bây giờ khi tôi chạy lệnh đó, công việc của chúng ta sẽ bắt đầu thử nghiệm đối với hai phiên bản dịch vụ. Và bạn sẽ thấy đầu ra xuất hiện ở phía bên tay phải ở đây ngay. Được rồi, chúng tôi bắt đầu, điểm chuẩn của chúng tôi. Và chúng tôi có tổng số 1.000 yêu cầu. Điều đó có vẻ nhanh hơn đáng kể. Bây giờ ở phần cao cấp, chúng tôi có một số yêu cầu mất nhiều thời gian hơn, nhưng đó là phân vị thứ 99, phân vị thứ 98. Đó có lẽ chỉ là một quang sai. Nếu bạn nhìn vào phân vị thứ 50, chúng ta đang nói 646 mili giây. Và không phải để khoe khoang, nhưng nếu tôi kéo các dự đoán của mình lên, tôi có 670 mili giây, vì vậy chỉ nhanh hơn một chút. Thành thật mà nói, ước tính chủ yếu là do may mắn, nhưng đó đại khái là ước tính bạn có thể thực hiện về một sân bóng, tất nhiên, giả sử, bạn biết những điểm nghẽn của mình thực sự là gì. Hãy mở rộng quy mô lên đến năm phiên bản dịch vụ của chúng tôi. Và nếu bạn muốn, đó là trong hướng dẫn, hoặc bạn có thể chỉ cần kéo lên lệnh mà chúng tôi đã chạy trước đó. Đảm bảo rằng bạn có node ‑‑ bản sao - tối đa - mỗi - node. Mặc dù vậy, tôi nên nói, nếu bạn đặt điều này lần cuối, nó sẽ vẫn ở đó lần này. Hãy nhớ rằng, khi chúng tôi cập nhật các dịch vụ, chúng tôi chỉ thay đổi các tham số mà chúng tôi truyền vào. Vì vậy, chỉ trong trường hợp bạn quên lần đầu tiên, hãy bao gồm nó lần này. Đặt giá trị này là 5 và bởi vì chúng tôi có 5 chiếc bánh nướng, nên đây thực sự sẽ là một công việc toàn cầu. Được rồi, điều đó nói rằng dịch vụ của chúng tôi đang chạy trên tất cả năm node và chúng tôi có thể xác nhận điều đó tại đây. Chúng tôi có tất cả năm node của chúng tôi có một vùng chứa. Vì vậy, trở lại đây tại dòng lệnh, bây giờ, tôi cần làm gì tiếp theo như một phần của việc kích hoạt kiểm tra? Vì vậy, chúng tôi muốn chạy lại cùng một kịch bản, có nghĩa là chúng tôi chỉ muốn bắt buộc cập nhật lại công việc của mình. Vì vậy, hãy làm điều đó, và bạn sẽ thấy kết quả đầu ra ở phía bên tay phải , và vượt qua các ngón tay của chúng tôi rằng điều này diễn ra nhanh hơn, và vâng, nó đang di chuyển nhanh hơn trước đây. Và nhìn vào đó, ở phân vị thứ 50, bạn có nhìn vào đó không? Chúng tôi đã có 256. Ước tính của tôi một lần nữa là gì? 270. Bây giờ đây chỉ là một kịch bản giả do tôi nghĩ ra, nhưng tôi hy vọng bạn có thể thấy cách bạn có thể dễ dàng sử dụng điều này để kiểm tra các giả thuyết khác mà bạn có thể có. Ví dụ: nếu bạn cho rằng đó là node thắt cổ chai phía máy khách , thì bạn có thể muốn chia tỷ lệ phía kiểm tra hoặc phía công việc của phương trình chứ không phải chia tỷ lệ phía máy chủ hoặc có thể là sự kết hợp của cả hai hoặc có thể là node cổ chai mạng. Có nhiều cách khác nhau để bạn có thể triển khai các công việc và dịch vụ, các cách kết hợp khác nhau của chúng, tăng số lượng node bạn có, có thể là động bằng cách mở rộng quy mô vào đám mây và bạn có thể học được nhiều điều mà bạn có thể sẽ gặp khó khăn học chỉ với các máy ảo trên một máy duy nhất trong đó các máy ảo đó bị giới hạn ở cùng dung lượng mà máy có.

Tiêm cấu hình ứng dụng và bí mật trong thời gian chạy

Cấu hình và bí mật là giá đỡ mục đích đặc biệt!

Chủ đề cuối cùng mà tôi muốn đề cập thực sự liên quan đến một số tài nguyên mà chúng tôi chưa giải quyết mà chúng tôi có thể thêm vào các tệp ngăn xếp của chúng tôi, một phần của chức năng **Swarm**. Chúng tôi đã thấy các mạng, chúng tôi đã thấy các dịch vụ, chúng tôi đã thấy khối lượng thậm chí hoặc thậm chí liên kết gắn kết với máy chủ. Bây giờ, chúng ta sẽ xem xét các cấu hình và bí mật. Bạn có thể coi các tệp cấu hình giống như một tập tin, trên thực tế, nó có cùng một khái niệm, nó chỉ là một phiên bản đặc biệt của nó được điều chỉnh cho phù hợp với các khía cạnh độc đáo của việc định cấu hình một ứng dụng và cách bạn cần cập nhật các tệp cấu hình đó. Và sau đó, bạn cũng có thể nghĩ đến bí mật, ví dụ, các tệp chính, như cấu hình nhạy cảm hoặc khối lượng cần được ánh xạ vào vì chúng chỉ là tệp và do đó, có các chi tiết triển khai đặc biệt để giữ an toàn cho những bí mật đó. Nếu không, cấu hình, bí mật và thậm chí cả khối lượng có gắn kết, tất cả đều thực sự giống nhau vào cuối ngày. Và trên dòng lệnh, nếu bạn nhìn vào lệnh bí mật của **docker**, đây là lệnh quản lý với một tập hợp các lệnh phụ để giúp bạn quản lý các bí mật và tương tự, có lệnh cấu hình **docker** và một lần nữa, đó là lệnh quản lý và nó có các lệnh phụ để giúp bạn quản lý các tệp cấu hình. Và về các tệp cấu hình, chỉ cần nghĩ đến bất kỳ lúc nào bạn có một ứng dụng trong đó bạn có một số loại cấu hình XML hoặc JSON cần được đưa vào, cũng hợp lý để có một vòng đời riêng biệt so với dữ liệu liên tục, v.v. thực sự tuyệt vời khi có một bộ chức năng riêng biệt xoay quanh việc quản lý các cấu hình độc lập với việc chỉ liên tục với khối lượng, mặc dù bạn có thể giải quyết chúng bằng cách kiên trì với khối lượng giống như bạn có thể làm với bí mật, nhưng bí mật cần được giữ an toàn, vì vậy đó là lý do tại sao bạn có những triển khai đặc biệt này.

Sử dụng bí mật bên ngoài để tránh mật khẩu mã hóa cứng trong tệp ngăn xếp

Đầu tiên, tôi muốn nói về bí mật. Bây giờ tôi đang ngồi bên trong thư mục DB bên trong các ví dụ về ngăn xếp kho lưu trữ, và bên trong đây là hai ngăn xếp riêng biệt. Tôi muốn bắt đầu bằng cách chạy một khác biệt của hai ngăn xếp này. Cả hai ngăn xếp này tạo ra một phiên bản của máy chủ MySQL. Sự khác biệt là ở cách cấu hình mật khẩu gốc. Ở phía bên trái , chúng tôi có một mật khẩu được mã hóa cứng cho người dùng MySQL và ở phía bên phải , chúng tôi có một cách xác định mật khẩu mới. Chúng ta sẽ sử dụng khái niệm bí mật và để làm được điều đó, chúng ta phải thiết lập một số thứ bên trong tệp Soạn của chúng ta. Vì vậy, có, điều này hơi dài dòng hơn một chút, nhưng tôi sẽ cho bạn thấy nó đáng giá. Vì vậy, sự khác biệt đầu tiên mà chúng tôi có, MYSQL\_ROOT\_PASSWORD và sau đó là \_FILE, đây là một biến môi trường, chúng tôi đã chỉ ra / run / secret / db\_pass. Bây giờ, điều này đến từ đâu? Chà, tôi đã đề cập đến những bí mật cũng giống như những quyển sách, về cơ bản là giống nhau. Bây giờ để điều này được gắn ở đây, chúng tôi phải định cấu hình dịch vụ của mình bằng cách nào đó, dịch vụ db của chúng tôi và nói với **Docker** **Swarm** rằng chúng tôi muốn có thể truy cập bí mật đó, còn được gọi là cấp quyền truy cập vào bí mật và chúng tôi làm điều đó với bí mật bên dưới cấu hình dịch vụ db, bên dưới phần tử dịch vụ cấp cao nhất, phần tử này có một loạt các bí mật và trong trường hợp của chúng ta, chúng ta chỉ có một db\_pass. Vì vậy, tên được sử dụng ở đây là tên của tệp, theo mặc định, có thể tùy chỉnh. Khi thêm điều này vào cấu hình dịch vụ của chúng tôi, chúng tôi đã cấp quyền truy cập vào bí mật của mình. Bây giờ chúng ta phải nói về cách bí mật đó được tạo ra. Có hai cách nổi bật mà bạn có thể tạo bí mật, một là bên ngoài tệp ngăn xếp, cách khác là bạn thực sự có thể tham chiếu một tệp có nội dung bí mật bên trong nó. Kiểu đó đánh bại quan điểm vì lúc đó tệp phải có sẵn trên hệ thống tệp và vì vậy chúng tôi quay lại hình vuông với việc mã hóa mật khẩu của chúng tôi thành một tệp. Có lẽ chúng tôi có thể bảo vệ nó dễ dàng hơn tệp Soạn của chúng tôi, nhưng nó vẫn là một tệp vào cuối ngày không được mã hóa hoặc bảo vệ theo bất kỳ cách nào. Vì vậy, tôi sẽ không đi theo cách tiếp cận đó. Thay vào đó, tôi sẽ thiết lập phần tử cấp cao nhất bí mật , giống như các dịch vụ vì đây là hai loại tài nguyên riêng biệt. Vì vậy, khi thêm yếu tố cấp cao nhất bí mật này , tôi đang xác định một số bí mật sau đó có thể được sử dụng trong ngăn xếp của tôi, cụ thể là bởi các dịch vụ của tôi. Và sau đó tôi có d\_pass, cùng một tên, và tôi đã đặt bên ngoài thành true và điều đó có nghĩa là tôi cần phải tạo điều này bên ngoài ngăn xếp của mình và tôi vẫn chưa làm điều đó. Chúng ta sẽ đạt được điều đó trong giây lát. Tôi có một câu hỏi dành cho bạn. Bạn nghĩ điều gì sẽ xảy ra nếu chúng tôi cố gắng triển khai ngăn xếp này trước khi chúng tôi tạo bí mật? Hãy cùng tìm hiểu. Vì vậy, chúng tôi không thể tạo dịch vụ cho đến khi có sẵn bí mật. Chúng tôi đang nhận được một bí mật không tìm thấy ngoại lệ. Hãy lưu ý rằng mạng của chúng tôi được tạo mặc dù, mạng ngầm định hoặc mặc định, nhưng không phải là dịch vụ. Nếu tôi liệt kê các ngăn xếp của mình, bạn sẽ không thấy bất kỳ ngăn xếp nào được liệt kê, nhưng nếu tôi liệt kê các mạng của mình, bạn sẽ thấy mạng ngầm của chúng tôi có sẵn, đó là một trạng thái kỳ lạ. Vì vậy, chúng ta hãy làm điều này. Hãy loại bỏ ngăn xếp đó và điều đó sẽ loại bỏ mạng của chúng ta ít nhất. Vậy bây giờ bạn nghĩ chúng ta cần phải làm gì? Chúng ta cần tạo ra bí mật đó và cách tốt nhất để làm điều đó là sử dụng lệnh cấp cao nhất bí mật của **docker** , và đây cũng là lệnh **Swarm**. Nếu tôi chạy điều này mà không có args, bạn sẽ nhận được sự giúp đỡ. Và như bạn có thể thấy, chúng tôi có một vài lệnh phụ, một trong số đó được gọi là tạo. Và tất nhiên, nếu tôi chạy điều này mà không có bất kỳ đối số nào, tôi sẽ nhận được trợ giúp cho nó và lưu ý rằng nó yêu cầu tên của một bí mật và sau đó là một tệp hoặc tôi cần cung cấp dấu gạch ngang cho tiêu chuẩn. Vì vậy, chúng tôi Tôi sẽ tiếp tục và tạo db\_pass của chúng tôi, tôi sẽ đặt dấu gạch ngang ở đây có nghĩa là tiêu chuẩn, và sau đó tôi sẽ đến đây và có lẽ tôi sẽ chỉ đưa ra một số thông tin, một cái gì đó siêu bí mật, và bạn xem. Chúng tôi đã tạo ra bí mật của mình và chúng tôi nhận lại một ID cho bí mật. Vì vậy, bây giờ chúng ta có thể chạy lệnh ls bí mật. Và như bạn có thể thấy, chúng tôi có một db\_pass bí mật duy nhất, ID đó khớp với ID chúng tôi lấy lại ở đây và nếu bạn muốn, chúng tôi có thể kiểm tra bí mật của chúng tôi và về cơ bản chúng tôi sẽ lấy lại cùng một thông tin, nhưng hãy lưu ý, chúng tôi không lấy lại giá trị của mật khẩu ở đây. Vì vậy, bây giờ tôi sẽ có thể tạo ngăn xếp của mình. Gọi bí mật đó cho ngăn xếp và bạn đi, chúng tôi đã có mạng và dịch vụ của chúng tôi được tạo. Hãy chờ đợi một giây và xem xét các nhiệm vụ liên quan đến ngăn xếp bí mật của chúng tôi và chúng tôi vẫn đang ở trạng thái chuẩn bị, mặc dù thành phố mong muốn của chúng tôi đang hoạt động, vì vậy đó là sự hòa giải trạng thái mong muốn. Sao bạn lại nghĩ như vậy? Nhiều khả năng là do chúng tôi chưa có sẵn image nên bị kéo xuống. Vì vậy, chúng ta hãy xem xét điều này một lần nữa, và bạn bắt đầu, chúng tôi hiện đang hoạt động. Vì vậy, tôi có thể chuyển sang w1, đó là node nơi vùng chứa của chúng ta đang chạy và tôi sẽ thực thi vùng chứa để chúng ta có thể xem xét xung quanh. Hoàn thành tab một cách lười biếng và chờ đợi, và chúng tôi đang ngồi bên trong hệ thống tệp.

**Docker** thực thi xung quanh Vùng chứa MySQL của chúng tôi để tìm hiểu về bí mật

Chúng tôi đang ngồi bên trong vùng chứa MySQL của mình, vì vậy chúng tôi có thể kết nối với cơ sở dữ liệu của mình, không cần mật khẩu và bạn có thể thấy Access bị từ chối. Và nếu tôi đặt dấu - p vào đây, tôi sẽ được nhắc nhập mật khẩu, vượt qua và chỉ như vậy, tôi có quyền truy cập vào phiên bản MySQL. Có thể kết xuất cơ sở dữ liệu, tạo cơ sở dữ liệu, tôi sẽ gọi đây là foo, và sau đó tôi sẽ kết xuất cơ sở dữ liệu để xem điều đó có được cam kết hay không, và nhìn vào đó, nó đã hoạt động. Chúng tôi có một cơ sở dữ liệu foo, vì vậy chúng tôi có đặc quyền root đối với phiên bản MySQL của chúng tôi thông qua việc sử dụng bí mật. Bây giờ, chúng ta hãy xem bí mật đó được tạo ra như thế nào. Tôi sẽ thoát khỏi máy khách MySQL và tôi đang ngồi đây trong hệ thống tệp của vùng chứa của mình, và bạn nghĩ chúng ta nên xem xét điều gì để tìm ra bí mật và xem xét giá trị của nó? Chà, tôi đã đề cập rằng có một mount được cung cấp, vì vậy tôi muốn xem qua lệnh mount và liệt kê ra các mount bên trong vùng chứa này. Và tôi làm điều đó, và tôi tìm kiếm từ bí mật. Tôi tìm thấy một bản ghi phù hợp với bản ghi liên kết đến tệp run / secret / db\_pass của chúng tôi. Lưu ý rằng đây là một tmpfs mount. Nó xảy ra ở chế độ chỉ đọc , có nghĩa là chúng tôi không thể thực hiện bất kỳ thay đổi nào đối với nó. Đối với quan điểm, tmpfs là một nơi lưu trữ tệp tạm thời. Đây là điều tăng thêm mức độ bảo mật và tôi nên chỉ ra rằng đây là một chi tiết triển khai, một cái gì đó như thế này có thể dễ dàng thay đổi, nhưng điều quan trọng cần lưu ý là trên Linux, đây là một trong những cơ chế để bảo mật bí mật của bạn, chẳng hạn. , chúng không chỉ được sao chép vào hệ thống tệp của vùng chứa của bạn. Nhưng trong trường hợp này, chúng được gắn kết hoặc liên kết với hệ thống tệp của bạn, vì vậy chúng có thể truy cập thuận tiện thông qua các lệnh gọi hệ thống tệp thông thường. Và có lẽ tôi chỉ nên nói một cách tổng quát, ý tưởng với những bí mật, bởi vì về cơ bản chúng là một khối, chúng là một vật gắn kết, điều khiến chúng trở nên đặc biệt là chúng được bảo vệ khi vận chuyển và ở trạng thái nghỉ ngơi. Vì vậy, chúng ta hãy xem tập tin này. Vì vậy, trước tiên hãy xem run / secret. Bạn có thể thấy db\_pass là tệp duy nhất bên trong đó. Hãy xóa nội dung của nó ra và nhìn vào đó, chúng ta đã có mật khẩu mà tôi vừa đặt để sử dụng. Vì vậy, đó là cách MySQL truy cập vào bí mật của chúng tôi, đó là mật khẩu của chúng tôi cho cơ sở dữ liệu của chúng tôi. Bởi vì đây là một tệp, nó có thể không chỉ là một mật khẩu. Đây có thể là một chứng chỉ, có lẽ đó là một khóa ssh. Về cơ bản, bất kỳ cấu hình nhạy cảm nào của ứng dụng của bạn đều có thể được lưu trữ và nên được lưu trữ bí mật. Còn một thứ nữa mà tôi muốn xem bên trong thùng hàng này. Và tôi sẽ xóa màn hình ở đây và liệt kê nội dung. Đây là thư mục làm việc, khi tôi exe vào, bạn có thể thấy có entrypoint.sh. Nhiều image chính thức có một cái gì đó như thế này. Chúng chặn các đối số được truyền khi một vùng chứa được tạo và đôi khi bật chức năng, có thể chạy các lệnh khác nhau hoặc trong trường hợp của chúng ta, thực sự có mã bên trong entrypoint này, hãy mở nó ra để tìm \_file trong đây, có mã trong đây để điều chỉnh một số biến môi trường nổi bật thông qua phương thức file\_env này. Điều này điều chỉnh một số biến môi trường như mật khẩu người dùng gốc MySQL, để thay vào đó, chúng tôi có thể sử dụng một biến môi trường có đường dẫn đến bí mật của chúng tôi. Đây là một quy ước phổ biến và được đề xuất mà bạn sẽ thấy trong rất nhiều image chính thức và không chính thức và có lẽ bạn nên cân nhắc cho image của riêng mình. Và xóa màn hình của tôi, và tôi sẽ để vùng chứa ở đây. Tiếp theo, hãy nói về việc loại bỏ các bí mật.

Xóa bí mật và phần giới thiệu về cập nhật bí mật với liên kết đến chuyến đi sâu hơn trong ấn bản đầu tiên của khóa học này

Vì vậy, nếu chúng ta chạy lệnh bí mật của **docker** và xem các lệnh quản lý phụ, chúng ta có tạo, kiểm tra và ls. Chúng tôi đã bao gồm tất cả ba trong số đó. Chúng tôi chưa nói về việc xóa. Vì vậy, một vài điều, chúng tôi vẫn có ngăn xếp bí mật của chúng tôi và chúng tôi vẫn có bí mật của chúng tôi. Bạn nghĩ điều gì sẽ xảy ra nếu tôi cố gắng loại bỏ bí mật đó trước khi ngừng dịch vụ đang sử dụng nó? Vì vậy, chúng tôi đang sử dụng **docker** secret rm, và sau đó chúng tôi cần cung cấp db\_pass. Bạn nghĩ điều gì sẽ xảy ra khi tôi chạy nó? Nó sẽ thất bại. Bạn thậm chí có thể thấy rằng trong đầu ra, nó cho chúng ta biết những gì vẫn đang sử dụng db\_pass bí mật, cái mà chúng tôi đang cố gắng loại bỏ. Vì vậy, nếu chúng tôi gỡ bỏ dịch vụ của mình trước, hãy giả sử rằng chúng tôi đã hoàn thành việc đó, vì vậy chúng tôi đánh dấu ngăn xếp bí mật của mình. Bây giờ, bạn nghĩ điều gì sẽ xảy ra nếu tôi cố gắng xóa bí mật db\_pass của chúng tôi? Nó hoạt động. Vì vậy, bây giờ nếu chúng ta nhìn vào các ngăn xếp của mình, chúng ta sẽ không có. Và nếu chúng ta nhìn vào bí mật của mình, thì không có bí mật nào cả. Điều đáng làm nổi bật là bạn không thể loại bỏ bí mật nếu nó đang được sử dụng. Điều đó nói rằng, bạn có thể lấy bí mật từ một dịch vụ cụ thể, vì vậy bạn có thể tạm thời xóa quyền truy cập vào dịch vụ đó và sau đó loại bỏ nó miễn là bạn thêm lại nội dung nào đó hoặc miễn là nó không cần bí mật đó nữa. Và khi tôi nói thêm lại, ý tôi là có lẽ bạn đang cập nhật bí mật thành một giá trị mới, nhân tiện, bạn không thể cập nhật ngay một bí mật, đó là lý do tại sao không có lệnh cập nhật bí mật của **docker**. Bạn phải tạo một bí mật được đặt tên mới và đó là vì cả tệp bí mật và tệp cấu hình đều không thay đổi. Điều đó không có nghĩa là bạn không thể thay đổi liên kết cho những gì bí mật đang được sử dụng bởi một dịch vụ hoặc một cấu hình bởi một dịch vụ, nó chỉ có nghĩa là bạn cần thực hiện một số loại xoay vòng và trong tài liệu dành cho bí mật, bạn ' Tôi sẽ tìm thấy một phần tuyệt vời ở đây về các bí mật luân phiên, nó liên quan đến việc đặt tên cho các bí mật của bạn sao cho chúng được tạo phiên bản, có lẽ bạn có một con số buộc tội ở cuối, có lẽ bạn đặt ngày cho chúng. Và sau đó nếu bạn muốn xem một ví dụ về điều này, hãy đến với ấn bản đầu tiên của tôi về khóa học này, dài hơn một chút. Chúng tôi đã cố gắng cắt bớt bản cập nhật này. Phiên bản đầu tiên của tôi là một khóa học chuyên sâu hơn là một khóa học bắt đầu. Không sao đâu. Rất nhiều người đã tìm thấy giá trị tuyệt vời trong đó. Ấn bản đầu tiên của tôi có một số ví dụ bổ sung mà bạn có thể muốn xem qua, một trong số đó là quy ước cập nhật bí mật mà tôi nói về kỹ thuật răng cưa này. Điều đó có thể hữu ích. Bạn cũng có thể tìm thấy các clip nhỏ khác trong suốt khóa học này mà bạn muốn xem một lần để bổ sung cho nội dung mới nhất mà chúng tôi có ở đây trong khóa học làm mới. Nếu bạn gặp khó khăn khi tìm ấn bản đầu tiên của tôi hoặc ấn bản đầu tiên của bất kỳ ai cho bất kỳ khóa học nào, trang tác giả có tất cả các bổ sung của khóa học với các liên kết trực tiếp để bạn không phải tìm kiếm chúng. Tôi cũng đặt các liên kết vào kho khóa học nơi có mã mẫu để bạn có thể truy cập cả hai.

Bí mật và cấu hình là những khái niệm song song - Bí mật chỉ là cấu hình nhạy cảm

Khi đến lúc bạn muốn sử dụng bí mật, đó là lúc bạn nên đọc qua rất nhiều tài liệu này. Vì vậy, điều cuối cùng tôi muốn làm là chỉ ra một số tài nguyên tốt ở đây để biết bí mật, và sau đó bên trong đây bạn sẽ tìm thấy các liên kết. Vì vậy, nếu bạn truy cập vào bất kỳ trang tài liệu nào mà tôi đang đề cập đến, bạn sẽ tìm thấy các liên kết đến các trang tài liệu khác mà tôi đang đề cập đến. Nếu bạn tìm kiếm tại đây để soạn thư, bạn sẽ thấy phần tham khảo Soạn tệp để biết chi tiết. Nếu chúng tôi mở nó trong một tab mới, đây là tài liệu tham khảo Soạn tệp phiên bản 3 và bạn có thể thấy chúng tôi đã được đưa đến mỏ neo để biết bí mật. Điều này đang cấp quyền truy cập trên cơ sở từng dịch vụ. Vì vậy, đó là nơi chúng tôi nói, Này, dịch vụ này có quyền truy cập vào bí mật này. Sau đó cũng có một liên kết ở đây để đưa bạn đến cấu hình bí mật cấp cao nhất. Đó là nơi chúng tôi thực sự đang ở cấp cao nhất xác định những bí mật mà sau đó chúng tôi có thể cấp quyền truy cập. Và giống như điều này tồn tại cho những bí mật, nếu bạn nhìn vào cùng một tham chiếu đặt cấu hình, bạn sẽ tìm thấy một neo cấu hình. Và nó cũng có liên kết đến, trong cùng một tài liệu, phần cấu hình cấp cao nhất. Vì vậy, thay vì chỉ lặp lại về cơ bản cùng một trình diễn cho bạn, tôi sẽ để nó như một thách thức đối với bạn trong việc khám phá các tệp cấu hình. Chúng tương đương với một bí mật không nhạy cảm . Xem nếu bạn không thể làm những gì chúng tôi đã làm với bí mật nơi chúng tôi vào vùng chứa, chúng tôi xem xét các mount, chúng tôi tìm thấy vị trí bí mật của chúng tôi hoặc trong trường hợp này là cấu hình của chúng tôi được gắn kết ở đâu và sau đó chúng tôi chỉ chơi với các lệnh quản lý đó để biết bí mật, tốt, hãy thử với các lệnh quản lý cấu hình. Và sau đó có thể có một tài nguyên khác cho các cấu hình mà tôi muốn chỉ ra. Đó là hướng dẫn giống như chúng tôi đã có với những bí mật. Tôi nghĩ rằng nó được liên kết với điều này. Vâng chính nó. Đây chỉ là một phần phụ của hướng dẫn. Có một dữ liệu cấu hình lưu trữ bằng cách sử dụng các cấu hình **Docker** giống như có một trong số chúng để quản lý dữ liệu nhạy cảm với các bí mật **Docker**. Đây là những tài liệu song song. Họ có thể sẽ đọc rất nhiều thứ giống nhau, nhưng họ sẽ xem xét các sắc thái của từng loại thú cưỡi, nếu bạn muốn. Tôi muốn nói rằng trước tiên hãy thử tự tìm hiểu một số điều này, chỉ nghĩ về những gì chúng tôi đã làm với những bí mật và xem liệu bạn có thể ánh xạ nó vào cấu hình hay không. Và sau đó hãy đến đây và đọc qua điều này nếu bạn không thể tìm thấy điều gì đó bởi vì có sự khác biệt nhỏ giữa hai điều này, chẳng hạn như nơi mọi thứ được gắn ở đâu. Và với điều đó đã nói, đây là phần cuối của khóa học. Tôi thực sự thích trình bày về chủ đề này vì **Docker** **Swarm** là một trong những công cụ yêu thích của tôi. Nó cực kỳ dễ sử dụng để tập hợp các máy lại với nhau. Như bạn có thể thấy trong suốt khóa học này, chúng tôi không phải thực hiện vô số cấu hình hoặc tạo chứng chỉ hoặc tất cả những điều điên rồ đó. Chúng tôi chỉ có thể đứng dậy và chạy. Thậm chí còn có Play with Doctor nếu bạn không muốn phải đối phó với việc xây dựng một cụm đơn giản trong một vài lệnh bạn phải thực hiện để làm điều đó. Thậm chí có thể thử sử dụng nó trong môi trường phát triển hoặc thậm chí là môi trường CI như một bước tiếp theo. Chắc chắn có giá trị thu được từ nó bởi vì bạn có thể mở rộng vượt quá khả năng của một máy trong cả hai bối cảnh đó và thực sự mở rộng ở cấp độ thực sự lớn, chẳng hạn như trong môi trường CI. Và sau đó, nếu bạn thích những gì bạn thấy, chỉ cần đưa nó đến quá trình sản xuất. Nếu bạn muốn liên hệ với tôi, tôi có một biểu mẫu trên trang web của mình. Có vẻ như tôi cần cập nhật nó để không quá nhỏ, nhưng bạn có thể đưa thông tin của mình vào đây và tôi sẽ liên hệ lại với bạn qua email. Và đừng quên, trang khóa học tồn tại. Hãy ra ngoài và thảo luận, thảo luận với tôi mà không có tôi, giúp người khác học hỏi và đặt câu hỏi của bạn và xem có ai có thể giúp bạn không. Và tôi hy vọng bạn thích khóa học này và hẹn gặp lại bạn vào lần sau.