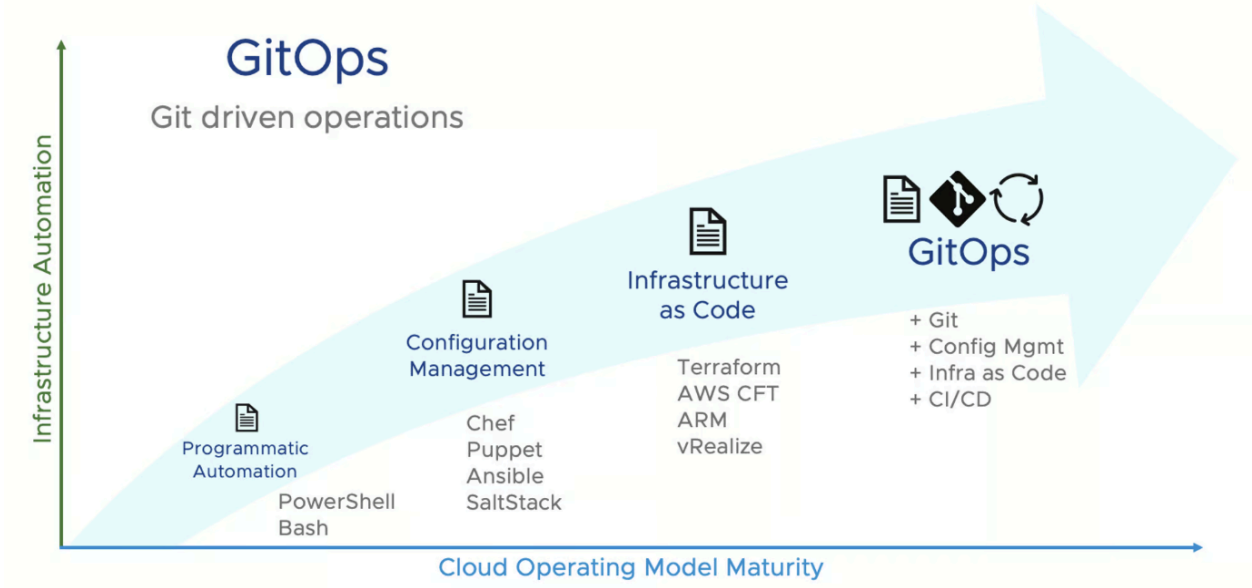
Tìm hiểu về GitOps

1. Các công cụ và công nghệ GitOps

Đặt vấn đề

Sự ra đời của Infrastructure as Code (IaC) giúp DevOps đơn giản hóa việc quản lý hệ thống cho cả đội DevOps và đội phát triển bằng cách sử dụng code để mô tả và cung cấp hệ thống.

Với IaC thì việc định nghĩa và tạo tất cả các tài nguyên cho môi trường từ Kubernetes, Terraform và Cloud đều được gói gọn trong các tệp tin code thay vì tạo bằng cách thủ công. Điều này đã giúp công việc định nghĩa hạ tầng từ cấu hình hệ thống, mạng và các tham số cho các môi trường khác nhau một cách rất dễ dàng.

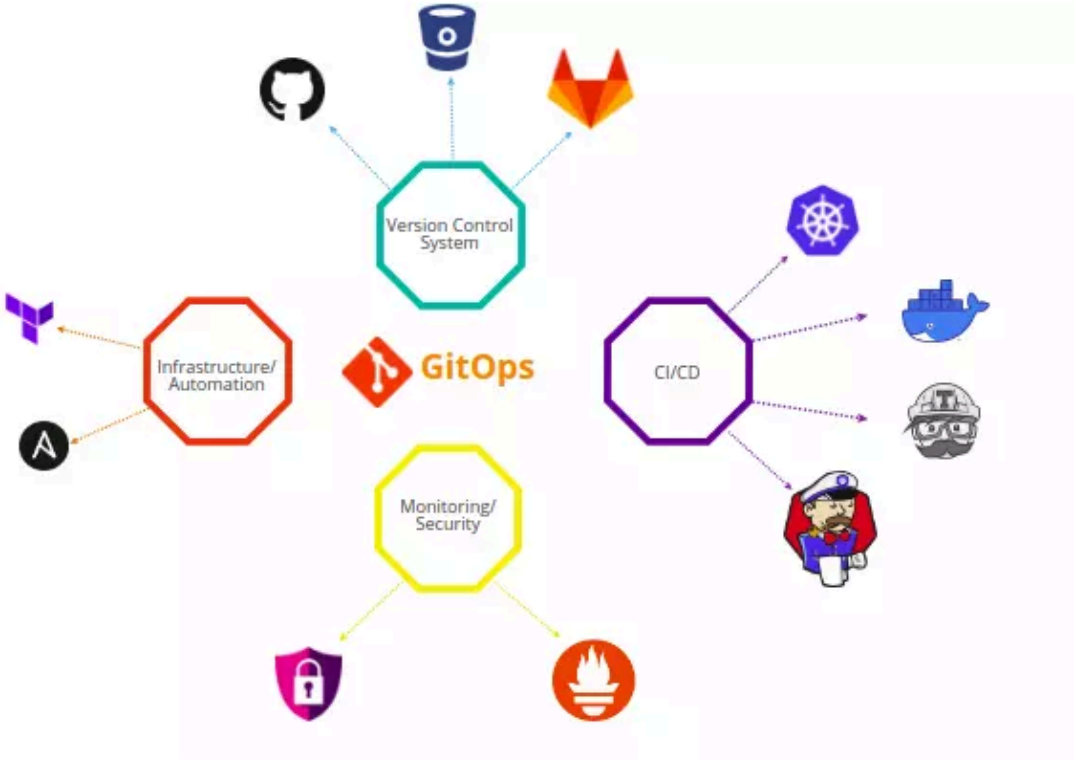


Tuy nhiên đa số các bạn DevOps thường để các tệp tin IaC ở dưới máy cá nhân và chạy thủ công để tạo hạ tầng, nếu có thay đổi gì ở hạ tầng các bạn chỉ sửa dưới máy của mình và chạy lại. Có một vài bạn kĩ hơn thì sẽ lưu các tệp tin IaC ở trên Git Repository để lưu trữ và theo dõi các thay đổi, nhưng vẫn chạy thủ công để thay đổi hạ tầng. Việc này sẽ

dẫn tới khó khăn trong việc quản lý khi đội của ta có nhiều người hơn. Do đó GitOps đã ra đời.

Với GitOps việc tạo, kiểm tra và thay đổi hạ tầng đều được thực hiện tự động thông qua việc sửa tệp tin IaC và đẩy lên Git. Lúc này đội của ta có thể dễ dàng theo dõi việc thay đổi hạ tầng.

GitOps là gì?



GitOps là một phương pháp tiếp cận quản lý và triển khai cơ sở hạ tầng và ứng dụng dựa trên Git. Mọi thay đổi đều được thực hiện thông qua các commit vào một repository Git, từ đó các công cụ GitOps sẽ tự động áp dụng các thay đổi này vào hệ thống thực tế.

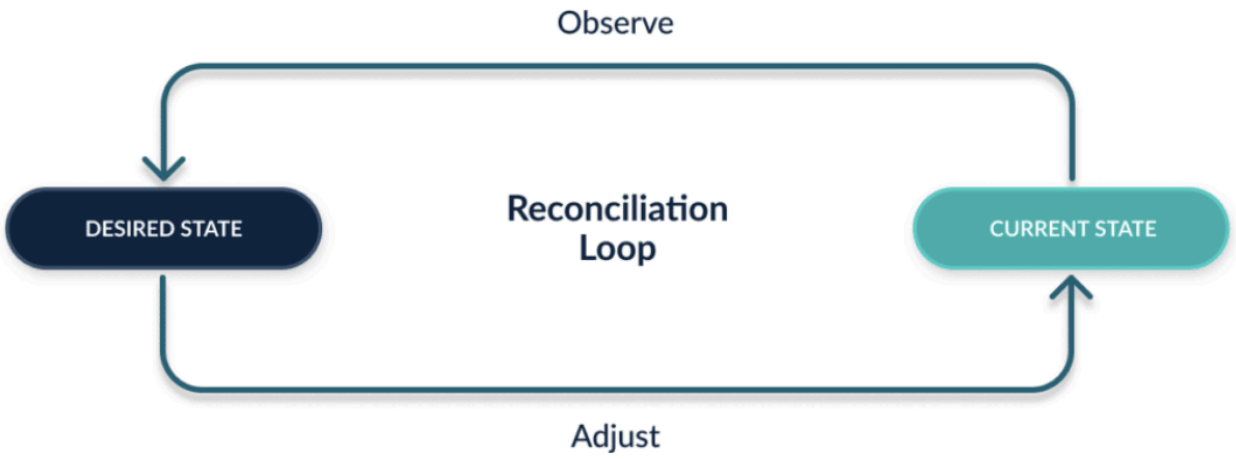
Một định nghĩa khác về GitOps là: GitOps là một phương pháp và thực hành sử dụng các kho lưu trữ Git như một nguồn sự thật duy nhất để cung cấp cơ sở hạ tầng như mã. Nó lấy các trụ cột và phương pháp từ văn hóa DevOps và cung cấp một khuôn khổ để bắt đầu hiện thực hóa các kết quả. Mối quan hệ giữa DevOps và GitOps rất gần gũi, vì

GitOps đã trở thành lựa chọn phổ biến để triển khai và nâng cao DevOps, kỹ thuật nền tảng, và SRE (Trích GitOps Cookbook).

GitOps là một phương pháp tiếp cận tập trung vào nhà phát triển, dựa trên các công cụ mà các nhà phát triển đã quen thuộc. Các nhà phát triển đã sử dụng Git cho mã nguồn của ứng dụng – GitOps mở rộng thực hành này sang cấu hình ứng dụng, cơ sở hạ tầng và các quy trình vận hành. GitOps lưu trữ mọi khía cạnh của cơ sở hạ tầng của dự án, bao gồm các tệp mã cơ sở hạ tầng như mã, tệp cấu hình và tệp mã ứng dụng, trong các kho lưu trữ Git. Mọi thay đổi đối với ứng dụng và cơ sở hạ tầng đều được đồng bộ hóa tự động với môi trường thực tế.

GitOps có thể được sử dụng để quản lý triển khai cho bất kỳ cơ sở hạ tầng nào. Nó đặc biệt hữu ích cho các nhà phát triển phần mềm và kỹ sư nền tảng làm việc với Kubernetes và muốn chuyển sang các mô hình hoạt động liên tục. GitOps giúp dễ dàng triển khai liên tục cho các ứng dụng gốc đám mây. Nó làm điều này bằng cách đảm bảo rằng cơ sở hạ tầng đám mây có thể tái tạo ngay lập tức dựa trên trạng thái của kho lưu trữ Git.

GitOps là một tập hợp các thực hành triển khai trong khi DevOps là một mô hình hoặc đúng hơn là một tư duy. Các nguyên tắc chung của chúng giúp các nhóm dễ dàng áp dụng quy trình GitOps cho các kỹ thuật DevOps hiện có.



4 Commandments của GitOps

1. Git for everything

GitOps mở rộng việc sử dụng Git cho mã nguồn của một ứng dụng sang cấu hình ứng dụng, cơ sở hạ tầng và các quy trình vận hành. Nó lưu trữ mọi khía cạnh của cơ sở hạ tầng dự án, bao gồm các tệp cơ sở hạ tầng như mã, tệp cấu hình và tệp mã ứng dụng, trong các kho lưu trữ Git, giúp quản lý triển khai phần mềm và cung cấp cơ sở hạ tầng dễ dàng hơn.

Bằng cách tận dụng định dạng khai báo, bạn có thể mô tả cách cơ sở hạ tầng của bạn hoạt động cũng như các ứng dụng đang chạy trên đó. Làm điều này cho phép theo dõi các thay đổi được thực hiện đối với bất kỳ môi trường nào và cho phép khôi phục lại, khôi phục, và các thuộc tính tự phục hồi bằng cách sử dụng bất kỳ hệ thống kiểm soát nguồn nào. Hãy chuyển từ các kịch bản lệnh ad-hoc bắt buộc sang cấu hình khai báo ở mọi cấp độ (ứng dụng và cơ sở hạ tầng).

2. V for Versioning

Các mô tả khai báo được lưu trữ trong kho lưu trữ hỗ trợ tính bất biến, phiên bản hóa và lịch sử phiên bản. Ví dụ, việc sử dụng Git cho các khai báo được đề cập ở trên cho phép bạn có một nơi duy nhất từ đó mọi thứ cho ứng dụng của bạn được suy ra và điều khiển. Điều này cho phép bạn dễ dàng xác định bất kỳ thay đổi nào được thực hiện bất kỳ lúc nào. Đừng cố gắng hiểu sự khác biệt giữa hai môi trường bằng tay. Chỉ cần xem tất cả các thay đổi được tìm thấy trong lịch sử kiểm soát phiên bản, đảm bảo rằng nền tảng luôn phù hợp với những gì được mô tả ở đó.

3. Auto-Pull

Sử dụng GitOps có nghĩa là bạn sử dụng các tác nhân phần mềm luôn chạy trong cụm, tự động kéo trạng thái từ Git theo các khoảng thời gian đều đặn và kiểm tra nó với trạng thái cụm trực tiếp. Bằng cách này, bạn luôn biết liệu phiên bản trong Git có giống với trạng thái trực tiếp hay không.

4. Loop the Loop

Hoạt động trong một vòng lặp kín đảm bảo trạng thái mong muốn của hệ thống khớp với trạng thái đã khai báo. Đây là một trong những tính năng quan trọng nhất vì nó cung cấp

phản hồi cho phép bạn và nhóm của mình kiểm soát tốt hơn các hoạt động và quy trình làm việc của mình.

Khi được sử dụng với kho lưu trữ của bạn, các tác nhân phần mềm GitOps có thể thực hiện nhiều chức năng khác nhau, đảm bảo tính tự phục hồi. Các tác nhân tự động sửa chữa trong trường hợp có sự cố, thực hiện các quy trình kiểm tra chất lượng cho quy trình làm việc của bạn và bảo vệ chống lại lỗi của con người hoặc can thiệp thủ công.

Tại sao nên sử dụng GitOps?

Sử dụng các quy trình làm việc dựa trên Git mà các nhà phát triển đã quen thuộc, GitOps mở rộng các quy trình hiện có từ phát triển ứng dụng đến triển khai, quản lý vòng đời ứng dụng và cấu hình cơ sở hạ tầng.

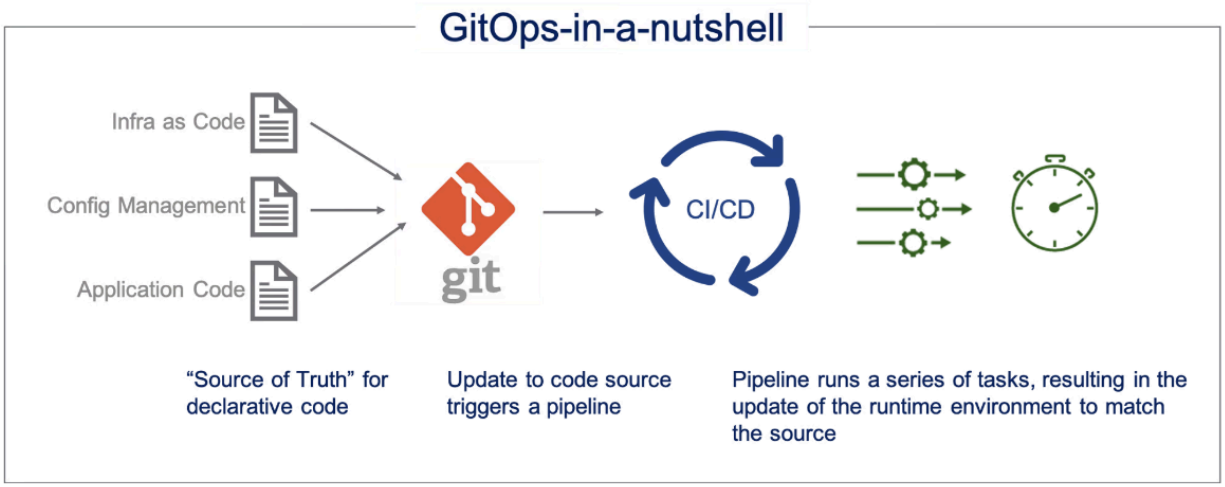
Mọi thay đổi trong suốt vòng đời ứng dụng đều được theo dõi trong kho lưu trữ Git và có thể kiểm tra được. Cách tiếp cận này mang lại lợi ích cho cả đội ngũ phát triển và đội ngũ vận hành vì nó tăng cường khả năng theo dõi và tái tạo các vấn đề một cách nhanh chóng, cải thiện an ninh tổng thể. Một điểm quan trọng là giảm rủi ro của các thay đổi không mong muốn (drift) và sửa chữa chúng trước khi chúng được đưa vào sản xuất.

Các lợi ích của GitOps:

- Quy trình làm việc tiêu chuẩn:
    - Sử dụng các công cụ quen thuộc và quy trình làm việc của Git từ các nhóm phát triển ứng dụng.
- Bảo mật nâng cao:
    - Xem xét các thay đổi trước, phát hiện sự lệch cấu hình và hành động kịp thời.
- Tính hiển thị và kiểm toán:
    - Ghi lại và theo dõi bất kỳ thay đổi nào đối với các cụm thông qua lịch sử Git.
- Tính nhất quán đa cụm:
    - Cấu hình một cách đáng tin cậy và nhất quán nhiều môi trường và nhiều cụm Kubernetes cũng như triển khai.

Cách hoạt động của GitOps

Cách hoạt động của GitOps đa số đều thông qua Git. Ví dụ, khi ta làm việc trong một nhóm DevOps lớn, bây giờ ta cần thay đổi cấu hình của hạ tầng. Đầu tiên ta sẽ kéo code trên Git xuống, thay đổi cấu hình của các tệp tin IaC mà liên quan tới hạ tầng. Sau khi thay đổi xong ta đẩy code lên trên Git, tạo yêu cầu merge vào nhánh chính. Sẽ có người xem xét và duyệt yêu cầu cho ta. Sau đó Git thông qua CI/CD sẽ cập nhật các thay đổi của hạ tầng và đảm bảo hạ tầng đồng nhất với các cấu hình của IaC. Nếu có bất kỳ lỗi nào ta có thể quay lại được thông qua cách *Revert* trên Git.



2. Các công cụ và công nghệ GitOps

Flux

Flux là một công cụ GitOps mã nguồn mở, được phát triển bởi Weaveworks. Flux tự động đồng bộ hóa trạng thái của Kubernetes cluster với trạng thái được định nghĩa trong một repository Git. Flux liên tục theo dõi các thay đổi trong repository Git và áp dụng những thay đổi đó vào cluster, đảm bảo rằng trạng thái của hệ thống luôn khớp với trạng thái mong muốn được lưu trữ trong Git. Flux hỗ trợ triển khai tự động, rollback, và quản lý secrets, giúp tăng cường tính nhất quán và an toàn cho quá trình triển khai.

Các tính năng chính của Flux:

- Triển khai tự động: Tự động áp dụng các thay đổi trong repository Git vào Kubernetes cluster.
- Quản lý secrets: Hỗ trợ quản lý và bảo vệ các secrets được lưu trữ trong Git.
- Khả năng rollback: Dễ dàng rollback về trạng thái trước đó nếu có vấn đề xảy ra.

ArgoCD

ArgoCD là một công cụ GitOps mã nguồn mở, được phát triển bởi Intuit. ArgoCD cung cấp giao diện người dùng trực quan và khả năng kiểm soát mạnh mẽ, cho phép quản lý và triển khai các ứng dụng Kubernetes từ Git. ArgoCD theo dõi các repository Git để phát hiện các thay đổi và tự động đồng bộ hóa những thay đổi đó với Kubernetes cluster. ArgoCD hỗ trợ nhiều chiến lược triển khai khác nhau và tích hợp tốt với các công cụ CI/CD khác.

Các tính năng chính của ArgoCD:

- Giao diện người dùng trực quan: Cung cấp giao diện người dùng dễ sử dụng để quản lý và giám sát các ứng dụng.
- Đồng bộ hóa tự động: Tự động đồng bộ hóa trạng thái của cluster với trạng thái được định nghĩa trong Git.
- Chiến lược triển khai linh hoạt: Hỗ trợ nhiều chiến lược triển khai như blue-green deployment, canary release.

Jenkins X

Jenkins X là một công cụ mã nguồn mở, mở rộng Jenkins để hỗ trợ triển khai CI/CD cho các ứng dụng Kubernetes. Jenkins X tích hợp các nguyên tắc của GitOps, tự động hóa toàn bộ quy trình phát triển từ mã nguồn đến triển khai. Jenkins X sử dụng các pipeline được định nghĩa trong repository Git để quản lý và tự động hóa các quy trình CI/CD.

Các tính năng chính của Jenkins X:

- Tích hợp GitOps: Sử dụng Git để quản lý và đồng bộ hóa các pipeline CI/CD.
- Hỗ trợ đa ngôn ngữ: Hỗ trợ nhiều ngôn ngữ lập trình và khung công nghệ khác nhau.
- Tích hợp với các công cụ CI/CD khác: Dễ dàng tích hợp với các công cụ như Jenkins, Tekton, và Prow.

Kustomize

Kustomize là một công cụ mã nguồn mở để quản lý cấu hình Kubernetes. Kustomize cho phép người dùng xác định và tùy chỉnh các tệp YAML mà không cần phải sao chép và chỉnh sửa trực tiếp các tệp đó. Kustomize tích hợp tốt với các công cụ GitOps như Flux và ArgoCD, giúp quản lý cấu hình một cách hiệu quả và dễ dàng.

Các tính năng chính của Kustomize:

- Quản lý cấu hình: Cho phép tùy chỉnh và quản lý các tệp YAML mà không cần sao chép.
- Tích hợp GitOps: Hỗ trợ tích hợp với các công cụ GitOps như Flux và ArgoCD.
- Khả năng mở rộng: Dễ dàng mở rộng và tùy chỉnh cấu hình cho các môi trường khác nhau.

3. Triển khai GitOps cho Cơ Sở Hạ Tầng Như Mã (IaC)

Tổng quan về IaC

Infrastructure as Code (IaC) là phương pháp quản lý và cung cấp hạ tầng thông qua mã hóa. Với IaC, các cấu hình hạ tầng như máy chủ, mạng, và các dịch vụ khác được mô tả dưới dạng mã nguồn. Điều này giúp tự động hóa quá trình triển khai hạ tầng, giảm thiểu lỗi và tăng tính nhất quán. IaC cho phép các nhóm phát triển và vận hành làm việc cùng nhau hiệu quả hơn, bởi vì mọi thay đổi đều có thể được kiểm soát phiên bản và kiểm thử trước khi triển khai vào môi trường thực tế.

Công cụ IaC phổ biến

Terraform

Terraform là một công cụ IaC mã nguồn mở do HashiCorp phát triển. Terraform cho phép mô tả hạ tầng dưới dạng mã, có thể dễ dàng quản lý và kiểm soát phiên bản qua Git. Terraform hỗ trợ nhiều nhà cung cấp dịch vụ đám mây khác nhau như AWS, Azure, Google Cloud, và nhiều nhà cung cấp khác. Với Terraform, bạn có thể tạo, thay đổi và cải thiện cơ sở hạ tầng một cách an toàn và có thể dự đoán được.

Các tính năng chính của Terraform:

- Mô tả hạ tầng dưới dạng mã: Sử dụng tệp cấu hình để mô tả các thành phần hạ tầng.
- Quản lý kiểm soát phiên bản: Cho phép kiểm soát phiên bản và quản lý các thay đổi trong Git.
- Hỗ trợ đa nền tảng: Tích hợp với nhiều nhà cung cấp dịch vụ đám mây và các công cụ khác.

Pulumi

Pulumi là một công cụ IaC mới, cho phép sử dụng các ngôn ngữ lập trình phổ biến như JavaScript, TypeScript, Python, Go và C# để mô tả hạ tầng. Pulumi cung cấp khả năng tích hợp tốt với các hệ sinh thái phát triển phần mềm hiện có, cho phép các nhà phát triển sử dụng các ngôn ngữ lập trình mà họ đã quen thuộc để quản lý hạ tầng.

Các tính năng chính của Pulumi:

- Sử dụng ngôn ngữ lập trình phổ biến: Cho phép mô tả hạ tầng bằng các ngôn ngữ lập trình như JavaScript, TypeScript, Python, Go và C#.
- Tích hợp với hệ sinh thái phát triển phần mềm: Dễ dàng tích hợp với các công cụ và quy trình phát triển phần mềm hiện có.
- Hỗ trợ đa nền tảng: Tương tự như Terraform, Pulumi cũng hỗ trợ nhiều nhà cung cấp dịch vụ đám mây.

Tích hợp GitOps với IaC

Tích hợp GitOps với IaC cho phép quản lý và triển khai hạ tầng một cách tự động và nhất quán. Mọi thay đổi trong repository Git sẽ được các công cụ GitOps tự động áp dụng vào hệ thống thực tế. Quá trình này đảm bảo rằng trạng thái của hệ thống luôn khớp với trạng thái mong muốn được lưu trữ trong Git.

Các bước tích hợp GitOps với IaC:

1. Mô tả hạ tầng dưới dạng mã: Sử dụng Terraform hoặc Pulumi để mô tả cấu hình hạ tầng dưới dạng mã.
2. Lưu trữ mã trong repository Git: Lưu trữ các tệp cấu hình hạ tầng trong repository Git để dễ dàng kiểm soát phiên bản và quản lý thay đổi.

3. Tự động hóa triển khai: Sử dụng các công cụ GitOps như Flux hoặc ArgoCD để theo dõi repository Git và tự động đồng bộ hóa các thay đổi với hệ thống thực tế.
4. Giám sát và kiểm soát: Sử dụng các công cụ giám sát và kiểm soát để đảm bảo rằng trạng thái của hệ thống luôn khớp với trạng thái được định nghĩa trong Git.

Ví dụ về quy trình tích hợp GitOps với IaC:

1. Tạo cấu hình hạ tầng với Terraform

```
provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
}
```

2. Lưu trữ cấu hình trong Git:
    - Lưu tệp cấu hình main.tf vào repository Git.
    - Commit và push các thay đổi lên repository Git.
3. Cài đặt và cấu hình Flux
    - Cài đặt Flux vào Kubernetes cluster.
    - Cấu hình Flux để theo dõi repository Git và tự động áp dụng các thay đổi.
4. Triển khai và giám sát:
    - Flux sẽ tự động đồng bộ hóa cấu hình hạ tầng từ repository Git với hệ thống thực tế.
    - Sử dụng các công cụ giám sát như Prometheus và Grafana để giám sát trạng thái của hệ thống.

4. Xu hướng phát triển trong tương lai

Xu hướng mới trong GitOps và IaC

Tự động hóa toàn diện

Các công cụ GitOps ngày càng được phát triển để tích hợp sâu hơn với các hệ thống tự động hóa, từ việc quản lý hạ tầng đến triển khai ứng dụng. Điều này không chỉ giúp tối ưu hóa quy trình triển khai mà còn đảm bảo tính nhất quán và giảm thiểu lỗi do sự can thiệp của con người. Các xu hướng tự động hóa toàn diện bao gồm:

- Công cụ CI/CD mạnh mẽ hơn: Các công cụ như Jenkins, GitLab CI, và CircleCI tiếp tục phát triển để tích hợp chặt chẽ hơn với GitOps, cung cấp các pipeline tự động hóa từ mã nguồn đến triển khai.
- Quản lý cấu hình tự động: Các công cụ như Ansible, Puppet, và Chef được tích hợp sâu hơn với GitOps, cho phép quản lý cấu hình tự động và đồng bộ hóa với trạng thái mong muốn trong Git.
- Hỗ trợ nhiều môi trường: Các công cụ GitOps hiện nay hỗ trợ nhiều môi trường khác nhau (on-premise, đám mây công cộng, đám mây riêng), giúp tự động hóa quy trình triển khai và quản lý hạ tầng trên các nền tảng khác nhau.

Khả năng mở rộng

Với sự phát triển của các hệ thống phức tạp và quy mô lớn, các công cụ GitOps và IaC cần phải được tối ưu hóa để xử lý các yêu cầu mở rộng. Các xu hướng chính bao gồm:

- Quản lý đa cluster: Các công cụ GitOps như ArgoCD và Flux hiện hỗ trợ quản lý nhiều Kubernetes cluster cùng một lúc, giúp dễ dàng quản lý và triển khai ứng dụng trên nhiều khu vực và môi trường khác nhau.
- Hỗ trợ đa nền tảng: Khả năng quản lý hạ tầng và ứng dụng trên nhiều nền tảng khác nhau (AWS, Azure, Google Cloud, on-premise) đang ngày càng được cải thiện, giúp các doanh nghiệp có thể dễ dàng mở rộng hệ thống của mình mà không gặp trở ngại về công nghệ.
- Tối ưu hóa hiệu suất: Các công cụ GitOps và IaC ngày càng được cải tiến để tối ưu hóa hiệu suất, giảm thiểu độ trễ và đảm bảo tính khả dụng cao cho các ứng dụng và dịch vụ.

Tích hợp AI/ML trong GitOps

AI và Machine Learning (AI/ML) đang được tích hợp vào GitOps để cung cấp các khả năng phân tích và tối ưu hóa nâng cao, giúp dự đoán và phát hiện lỗi sớm hơn, từ đó cải thiện hiệu quả và độ tin cậy của hệ thống. Một số ứng dụng chính của AI/ML trong GitOps bao gồm:

- Dự đoán lỗi: Sử dụng các thuật toán học máy để phân tích các mẫu và dự đoán các lỗi có thể xảy ra trong quá trình triển khai hoặc vận hành hệ thống. Điều này giúp giảm thiểu thời gian chết và tối ưu hóa hiệu suất hệ thống.
- Tối ưu hóa tài nguyên: AI/ML có thể được sử dụng để phân tích việc sử dụng tài nguyên và đề xuất các điều chỉnh để tối ưu hóa hiệu suất và giảm thiểu chi phí. Ví dụ, hệ thống có thể tự động điều chỉnh số lượng instance dựa trên lưu lượng truy cập và nhu cầu sử dụng.
- Tự động hóa phản ứng sự cố: Sử dụng AI/ML để phát hiện sớm các sự cố và tự động thực hiện các biện pháp khắc phục. Ví dụ, hệ thống có thể tự động khởi động lại các dịch vụ hoặc điều chỉnh cấu hình để khắc phục sự cố mà không cần sự can thiệp của con người.
- Tối ưu hóa quy trình CI/CD: AI/ML có thể phân tích các pipeline CI/CD để tìm ra các điểm nghẽn và đề xuất các cải tiến để tăng tốc độ triển khai và giảm thiểu thời gian chờ đợi.

5.Kết Luận

GitOps không chỉ là một phương pháp tiếp cận mới mẻ mà còn là một bước tiến quan trọng trong việc quản lý và triển khai cơ sở hạ tầng và ứng dụng. Với sự kết hợp của Git và IaC, GitOps mang lại nhiều lợi ích vượt trội như tính nhất quán, khả năng theo dõi và tự động hóa. Bằng cách áp dụng GitOps, các doanh nghiệp có thể tối ưu hóa quy trình, nâng cao hiệu quả làm việc và đảm bảo rằng hệ thống luôn hoạt động ổn định và hiệu quả. Việc triển khai GitOps không chỉ giúp giảm thiểu lỗi và rủi ro mà còn tạo ra một môi trường làm việc linh hoạt và đổi mới, mở ra nhiều cơ hội phát triển bền vững trong tương lai.

Tài liệu tham khảo

1. Weaveworks GitOps
2. Flux Documentation
3. Terraform Documentation
4. Jenkins Documentation
5. Kustomize Documentation
6. ArgoCD Documentation
7. Pulumi Documentation
8. AI/ML in DevOps