**Ảnh có chứa quảng trường

Mô tả được tạo tự độngĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA HỆ THỐNG THÔNG TIN**

**BÁO CÁO CHỦ ĐỀ 19**

**MÔN ĐIỆN TOÁN ĐÁM MÂY**

**Đề tài: TERRAFORM**



GVHD: ThS. Hà Lê Hoài Trung

Nhóm sinh viên thực hiện:

1. Nguyễn Cao Khoa MSSV: 19521694

2. Hồ Trọng Khang MSSV: 19521661

3. Trần Nhật Tân MSSV: 19522177

4. Huỳnh Quốc Khánh MSSV: 19521677

Tp. Hồ Chí Minh, 9/2022

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*……., ngày……...tháng……năm 2022…*

**Người nhận xét**

*(Ký tên và ghi rõ họ tên****)***

**LỜI CẢM ƠN**

Nhóm xin chân thành cảm ơn thầy Hà Lê Hoài Trung - giảng viên môn Điện Toán Đám Mây đã nhiệt tình chỉ bảo và hướng dẫn tận tình trong suốt thời gian thực hiện đồ án môn học. Nhờ đó, chúng em đã có được nhiều kiến ​​thức bổ ích trong việc ứng dụng cũng như kỹ năng làm đồ án.

Bắt đầu từ việc tìm hiểu về các nền tảng Cloud Computing. Dựa trên những kiến ​​thức được giáo viên cung cấp trên lớp, kết hợp với việc tự nghiên cứu các công cụ và kiến ​​thức mới, nhóm cố gắng thực hiện dự án một cách tốt nhất có thể. Trong thời gian thực hiện dự án, nhóm chúng em đã sử dụng kiến ​​thức nền tảng đã tích lũy được, kết hợp học tập và nghiên cứu kiến ​​thức mới. Từ đó, nhóm của chúng em sử dụng đầy đủ các thông tin thu thập được để đưa ra báo cáo dự án tốt nhất có thể. Tuy nhiên, trong quá trình thực hiện, nhóm chúng em không tránh khỏi những thiếu sót. Vì vậy, nhóm rất mong nhận được sự góp ý của các thầy để nhóm em hoàn thiện hơn kiến ​​thức và chuẩn bị cho các đề tài khác trong tương lai.

Sau cùng, nhóm xin kính chúc thầy thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp là truyền đạt kiến thức cho các bạn sinh viên.

Xin chân thành cảm ơn Thầy!

# DANH MỤC CÁC BẢNG, HÌNH ẢNH

**Danh mục bảng:**

[Bảng 1: So sánh Terraform & CloudFormation 11](#_Toc115669393)

**Danh mục ảnh:**

[Hình 1: Mô hình IaC 6](#_Toc115740936)

[Hình 2: Terraform quản lý tài nguyên thông qua API 8](#_Toc115740937)

[Hình 3: Quy trình hoạt động của Terraform 9](#_Toc115740938)

[Hình 4: Trang dowload Terraform 14](#_Toc115740939)

[Hình 5: Folder cài đặt Terraform vừa tải về 15](#_Toc115740940)

[Hình 6: Folder được giải nén và chuyển đến Home 15](#_Toc115740941)

[Hình 7: Câu lệnh mở file bashrc 16](#_Toc115740942)

[Hình 8: Tạo biến môi trường cho Terraform 16](#_Toc115740943)

[Hình 9: Thực thi file .bashrc 17](#_Toc115740944)

[Hình 10: Mở app để thay đổi môi trường 17](#_Toc115740945)

[Hình 11: Giao diện của System Properties 18](#_Toc115740946)

[Hình 12: Thêm đường dẫn Terra vào môi trường 18](#_Toc115740947)

[Hình 13: Kiểm tra cài đặt Terraform 19](#_Toc115740948)

**MỤC LỤC**

[**LỜI CẢM ƠN** 3](#_Toc115740961)

[DANH MỤC CÁC BẢNG, HÌNH ẢNH 4](#_Toc115740962)

[**MỤC LỤC** 5](#_Toc115740963)

[PHẦN 1: THÔNG TIN TỔNG QUAN 6](#_Toc115740964)

[**1.** **GIỚI THIỆU TỔNG QUAN VỀ INFRASTRUCTURE AS CODE** 6](#_Toc115740965)

[**1.1.** **Infrastructure as Code là gì** 6](#_Toc115740966)

[**1.2.** **Lợi ích khi sử dụng Infrastructure as Code** 7](#_Toc115740967)

[**2.** **GIỚI THIỆU TỔNG QUAN VỀ TERRAFORM** 7](#_Toc115740968)

[**2.1.** **Terraform là gì** 7](#_Toc115740969)

[**2.2.** **Lịch sử phát triển** 8](#_Toc115740970)

[**2.3.** **Terraform hoạt động như thế nào** 8](#_Toc115740971)

[**2.4.** **Terraform được dùng để làm gì** 10](#_Toc115740972)

[**2.5.** **Đặc điểm của Terraform** 10](#_Toc115740973)

[**2.6.** **So sánh Terraform & CloudFormation** 11](#_Toc115740974)

[**2.7.** **Lợi ích và thách thức khi sử dụng Terraform** 11](#_Toc115740975)

[**3.** **LÍ DO NÊN CHỌN TERRAFORM ĐỂ SỬ DỤNG** 12](#_Toc115740976)

[**3.1.** **Configuration Management & Provisioning** 12](#_Toc115740977)

[**3.2.** **Một số lí do khác** 13](#_Toc115740978)

[PHẦN 2: CÀI ĐẶT TERRAFORM 14](#_Toc115740979)

[**1.** **TẢI TERRAFORM** 14](#_Toc115740980)

[**2.** **CÀI ĐẶT MÔI TRƯỜNG** 16](#_Toc115740981)

[**2.1.** **Đối với Ubuntu** 16](#_Toc115740982)

[**2.2.** **Đối với Win** 17](#_Toc115740983)

[**3.** **KIỂM TRA CÀI ĐẶT** 19](#_Toc115740984)

[PHẦN 3: DEMO 20](#_Toc115740985)

[PHẦN 4: TÀI LIỆU THAM KHẢO 21](#_Toc115740986)

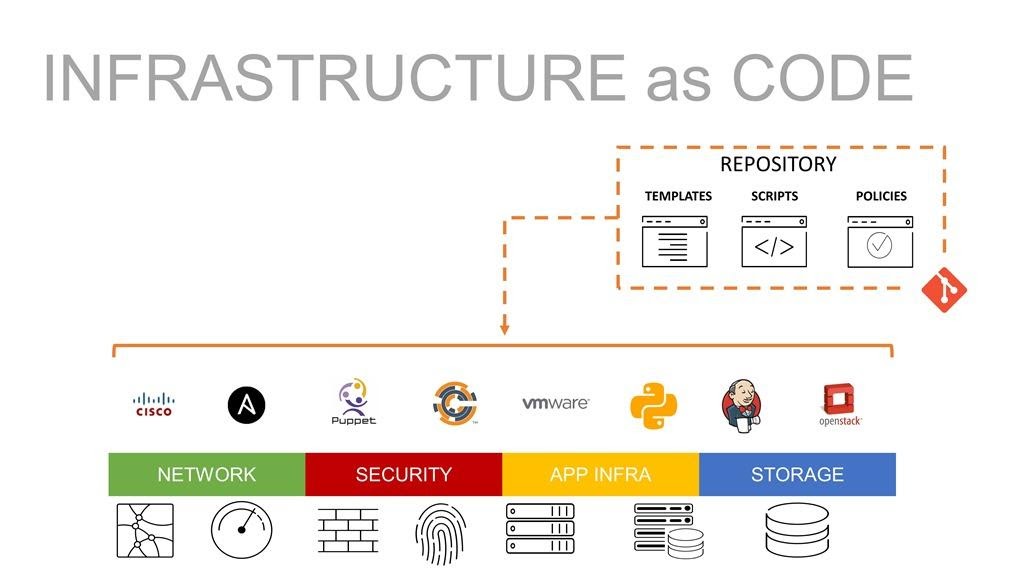
# PHẦN 1: THÔNG TIN TỔNG QUAN

1. **GIỚI THIỆU TỔNG QUAN VỀ INFRASTRUCTURE AS CODE**
   1. **Infrastructure as Code là gì**

Infrastructure as code - Cơ sở hạ tầng như mã code, còn được gọi là IaC, là một mô hình thiết lập và triển khai công nghệ trong đó các nhà phát triển hoặc nhóm vận hành tự động quản lý và cung cấp các gói công nghệ cho ứng dụng thông qua phần mềm, thay vì sử dụng quy trình thủ công để cấu hình các thiết bị phần cứng và hệ điều hành rời rạc như trước kia. Cơ sở hạ tầng dưới dạng mã đôi khi được gọi là cơ sở hạ tầng được lập trình hoặc cơ sở hạ tầng phần mềm xác định.

Hiện nay, việc quản lý các web applications khá là phức tạp, ví dụ bạn phải quản lý load balancer, vài web servers, database servers, ... Bạn có thể sẽ phải mất hàng giờ để provisioning và quản lý những hệ thống này. Infrastructure as Code có thể hiểu là bạn có thể thiết lập/quản lý những stack trước kia của hệ thống thông qua việc định nghĩa chúng trong 1 file script chẳng hạn thay vì tốn thời gian và công sức setup manual từng thứ.

Hay nói rõ hơn **Infrastructure as Code** là dùng nền tảng hoặc script để tự động hoá quá trình cấu hình cơ sở hạ tầng thay vì phải setup thủ công.



Hình 1: Mô hình IaC

* 1. **Lợi ích khi sử dụng Infrastructure as Code**

Khi sử dụng IaC người dùng và các nhà phát triển có những lợi ích sau:

* Cải thiện tốc độ: Tự động hóa nhanh hơn điều hướng thủ công một giao diện khi bạn cần triển khai hoặc kết nối tài nguyên.
* Cải thiện độ tin cậy: Nếu cơ sở hạ tầng của bạn lớn, bạn sẽ dễ dàng định cấu hình sai tài nguyên hoặc dịch vụ cung cấp theo thứ tự sai. Với IaC, tài nguyên luôn được cung cấp và cấu hình chính xác như đã khai báo.
* Ngăn trôi cấu hình: Lỗi cấu hình xảy ra khi cấu hình đã cung cấp cho môi trường của bạn không còn khớp với môi trường thực tế.
* Hỗ trợ thử nghiệm, kiểm tra và tối ưu hóa: Bởi vì Infrastructure as Code làm cho việc cung cấp cơ sở hạ tầng mới nhanh hơn và dễ dàng hơn rất nhiều, bạn có thể thực hiện và kiểm tra các thay đổi thử nghiệm mà không cần đầu tư nhiều thời gian và tài nguyên; và nếu bạn thích kết quả, bạn có thể nhanh chóng mở rộng quy mô cơ sở hạ tầng mới cho sản xuất.

1. **GIỚI THIỆU TỔNG QUAN VỀ TERRAFORM**
   1. **Terraform là gì**

Terraform là một công cụ mã nguồn mở hoàn toàn miễn phí được phát hành vào

tháng 7 năm 2014 bởi HashiCorp. Công cụ này giúp người dùng định nghĩa và lưu

trữ thông tin tài nguyên bên trong hạ tầng hệ thống của mình thông qua các file

code. Từ những file code này người dùng có thể sử dụng để triển khai hạ tầng của

mình trên cloud như AWS, GCP, Azure, Digital Ocean, GitHub, Cloudflare,… hay

cả VMware vSphere,…

Đây cũng là một trong những công cụ IaC (Infrastructure as code) phổ biến nhất

tính tới thời điểm hiện tại. Các công cụ khác có thể kể đến bao gồm: Ansible, Puppet, Vagrant, Chef, SaltStack,…

Bên cạnh đó Terraform có thể quản lý và hỗ trợ hầu hết các “service provider” đình đám hiện nay như: [Amazon Web Service](https://hungphamdevweb.com/quan-ly-hinh-anh-trong-wordpress-bang-amazon-s3.html), Google Cloud, Alibaba Cloud, Microsoft Azure.

* 1. **Lịch sử phát triển**
* Terraform là công cụ cơ sở hạ tầng bất biến đa đám mây được viết bằng Go.
* Thiết kế bởi Mitchell Hashimoto et al.
* Phát triển bởi HashiCorp.
* Phát hành lần đầu 28 tháng 7 năm 2014.
* Hỗ trợ các hệ điều hành: Linux, FreeBSD, macOS, OpenBSD, Solaris, and Microsoft Windows.
  1. **Terraform hoạt động như thế nào**

Terraform tạo và quản lý tài nguyên trên nền tảng đám mây và các dịch vụ khác thông qua giao diện lập trình ứng dụng (API) của họ. Các nhà cung cấp cho phép Terraform làm việc với hầu hết mọi nền tảng hoặc dịch vụ có API có thể truy cập được.

Diagram

Description automatically generated

Hình 2: Terraform quản lý tài nguyên thông qua API

HashiCorp và cộng đồng Terraform đã viết **hơn 1700 nhà cung cấp** để quản lý

hàng nghìn loại tài nguyên và dịch vụ khác nhau, và con số này tiếp tục tăng lên.

Bạn có thể tìm thấy tất cả các nhà cung cấp có sẵn công khai trên [Đăng kí](https://registry.terraform.io), bao

gồm Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP),

Kubernetes, Helm, GitHub, Splunk, DataDog,...

Quy trình làm việc Terraform cốt lõi bao gồm ba giai đoạn:

* **Write:** Bạn xác định các tài nguyên, có thể trên nhiều nhà cung cấp và dịch vụ

đám mây. Ví dụ: bạn có thể tạo cấu hình để triển khai ứng dụng trên máy ảo

trong mạng Đám mây riêng ảo (VPC) với các nhóm bảo mật và bộ cân bằng tải.

* **Plan:** Terraform tạo ra một kế hoạch thực thi mô tả cơ sở hạ tầng mà nó sẽ tạo, cập nhật hoặc phá hủy dựa trên cơ sở hạ tầng hiện có và cấu hình của bạn.
* **Apply:** Khi được phê duyệt, Terraform thực hiện các hoạt động được đề xuất

theo đúng thứ tự, tôn trọng bất kỳ sự phụ thuộc tài nguyên nào. Ví dụ: nếu bạn cập nhật các thuộc tính của một VPC và thay đổi số lượng máy ảo trong VPC đó, Terraform sẽ tạo lại VPC trước khi thay đổi quy mô của các máy ảo.

Diagram

Description automatically generated

Hình 3: Quy trình hoạt động của Terraform

* 1. **Terraform được dùng để làm gì**
* Quản lý tài nguyên bên ngoài: Terraform hỗ trợ cơ sở hạ tầng đám mây công cộng và [riêng tư](https://www.techtarget.com/searchcloudcomputing/definition/private-cloud), cũng như triển khai các thiết bị mạng và phần mềm dưới dạng dịch vụ ([SaaS](https://www.techtarget.com/searchcloudcomputing/definition/Software-as-a-Service)).
* Triển khai đa đám mây: Khả năng gốc của công cụ phần mềm để hỗ trợ nhiều dịch vụ đám mây giúp tăng khả năng chịu lỗi.
* Các ứng dụng nhiều tầng: Terraform cho phép mỗi bộ sưu tập tài nguyên dễ dàng được tăng hoặc giảm quy mô khi cần thiết.
* Các cụm tự phục vụ: Các cơ quan đăng ký giúp người dùng dễ dàng tìm thấy các cấu hình đóng gói sẵn có thể được sử dụng nguyên trạng hoặc sửa đổi để đáp ứng một nhu cầu cụ thể.
* Mạng do phần mềm định nghĩa ([SDN](https://www.techtarget.com/searchnetworking/definition/software-defined-networking-SDN)): Khả năng đọc của Terraform giúp các kỹ sư mạng dễ dàng hệ thống hóa cấu hình cho SDN.
* Trình lập lịch tài nguyên: Các mô-đun Terraform có thể dừng và bắt đầu tài nguyên trên AWS và cho phép Kubernetes lên lịch cho các bộ chứa Docker.
* Môi trường dùng một lần: Các mô-đun có thể được sử dụng để tạo một môi trường thử nghiệm đặc biệt, nhanh chóng cho mã trước khi nó được đưa vào sản xuất.
  1. **Đặc điểm của Terraform**

Terraform hoạt động bằng cách xây dựng một cơ sở dữ liệu đồ thị cung cấp cho

các toán tử cái nhìn sâu sắc về các phụ thuộc tài nguyên. Nó cũng tạo ra một kế

hoạch thực thi cho phép các nhà khai thác xem chuỗi các bước mà Terraform sẽ

thực hiện khi một cài đặt được áp dụng hoặc một thay đổi được thực hiện. Phần

mềm Terraform cũng bao gồm những điều sau đây:

* Một bảng điều khiển cho phép người dùng quan sát các chức năng như số, chuỗi, ngày và giờ, bộ sưu tập cũng như các chức năng mã hóa.
* Một ngôn ngữ cấu hình hỗ trợ nội suy và cho phép quản trị viên truyền các hàm dưới dạng một chuỗi để thực hiện một loạt các hoạt động.
* Khả năng dịch mã HCL sang định dạng [JSON](https://theserverside.techtarget.com/definition/JSON-Javascript-Object-Notation).
* Một tính năng được gọi là Module Count chỉ định số lượng mô-đun đã được áp dụng cho cơ sở hạ tầng.
  1. **So sánh Terraform & CloudFormation**

Bảng 1: So sánh Terraform & CloudFormation

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **Terraform** | **CloudFormation** |
| **Phạm vi** | Bao gồm **hầu hết** các tài nguyên và thành phần AWS và tương đối nhanh hơn trong việc hỗ trợ các tính năng AWS mới. | Bao gồm các thành phần **chính** của AWS nhưng chậm hơn trong việc hỗ trợ các chức năng AWS mới. |
| **Giấy phép** | Dự án mã nguồn mở. | Cung cấp miễn phí trên AWS. |
| **Hỗ trợ** | Hashicorp hỗ trợ 24/7. | Hỗ trợ bởi AWS. |
| **Quản lý các state** | Lưu trữ state trên đĩa và cho phép sử dụng từ xa. | Lưu trữ và quản lý state trong các stack. |
| **Mô-đun** | Cung cấp hoàn toàn hỗ trợ gốc cho các mô-đun. | Không hỗ trợ sử dụng mô-đun nhưng có tính năng mô-đun hóa các templates. |
| **Thay đổi xác minh** | Sử dụng lệnh ‘plan’ để xác định những thay đổi cần thiết. | Sử dụng ‘change sets’ để xác định yêu cầu hoặc thay đổi được thực hiện. |

* 1. **Lợi ích và thách thức khi sử dụng Terraform**

Những lợi thế của việc sử dụng Terraform bao gồm khả năng:

* Dịch mã HCL thành JSON.
* Hỗ trợ nhiều nền tảng đám mây.
* Thực hiện các thay đổi gia tăng đối với các nguồn lực.
* Cung cấp hỗ trợ cho mạng do phần mềm xác định.
* Nhập các tài nguyên hiện có vào trạng thái Terraform.
* Khóa các mô-đun trước khi áp dụng các thay đổi trạng thái để đảm bảo rằng chỉ một người có thể thực hiện thay đổi tại một thời điểm.

Tuy nhiên, có một số nhược điểm khi sử dụng Terraform:

* Các bản phát hành và bản cập nhật mới có thể có lỗi.
* Các state phải đồng bộ với cơ sở hạ tầng mọi lúc.
* Nếu người dùng không chọn sử dụng JSON, họ sẽ phải học một ngôn ngữ mới đó là HCL.
* Nó không có xử lý lỗi.
* Đổi tên tài nguyên và di chuyển chúng sâu hơn vào các mô-đun có thể khó khăn.

1. **LÍ DO NÊN CHỌN TERRAFORM ĐỂ SỬ DỤNG**

Khi chúng ta search “infrastructure-as-code” trên Google, nó sẽ liệt kê ra rất nhiều

tools khá phổ biến như: Chef, Puppet, Ansible, SaltStack, CloudFormation,

**Terraform**,..

Tất cả những tools trên đều có thể giúp bạn quản lý infrastructure (servers, vpc,

storage ...). Chúng hầu hết cũng là open source, được hỗ trợ bởi đông đảo

contributors và có thể dùng với nhiều cloud provider khác nhau như Google Cloud,

AWS, Azure, ... (CloudFormation thì không open source và chỉ hỗ trợ AWS).

Vậy thì **tại sao lại chọn Terraform ?** Dưới đây là một số lí do chính:

* 1. **Configuration Management & Provisioning**

Chef, Puppet, Ansible và SaltStack đều là **configuration management tool**, tức là nó được design để cài đặt và quản lý phần mềm trên những servers có sẵn. CloudFormation và Terraform là **provisioning tool**, có nghĩa là nó được thiết kế để xây dựng servers (cũng như việc thiết lập load balancers, databases, networking,...) và việc config server sẽ nhường lại cho tool khác.

2 khái niệm/công việc của những tool này không khác nhau hoàn toàn, chúng có thể làm được những phần việc mà cái kia cũng có thể làm được. Nhưng ở đây chúng ta sẽ tập trung vào việc configuration management hoặc provisioning sẽ phù hợp trong trường hợp nào.

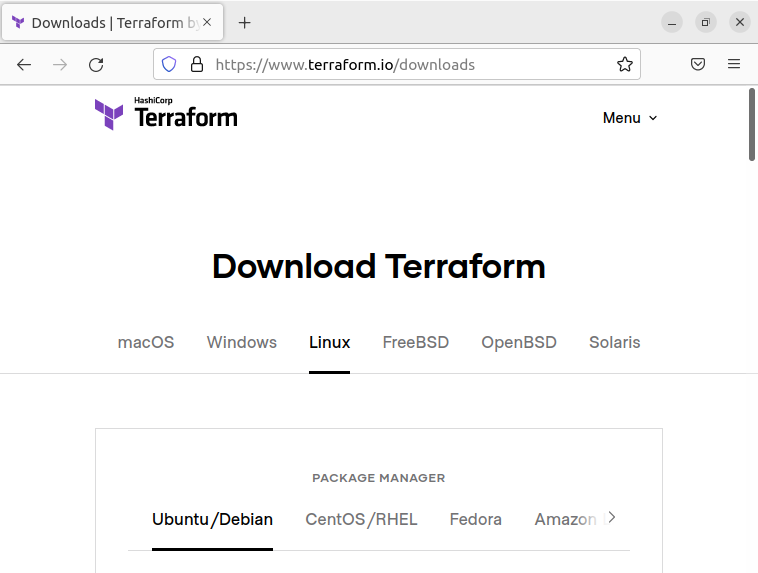
Ví dụ, bạn dùng Docker thì hầu hết phần việc config đã được thực hiện trước đó rồi. Trong Docker images thì hầu hết những phần mềm mà server bạn cần đều đã được cài đặt và config cả rồi. Một khi bạn đã có Docker images rồi thì điều bạn cần làm chỉ là chạy nó mà thôi. Điều bạn thực sự cần làm ở đây là provision những servers để chạy những container này, **do vậy Terraform sẽ phù hợp và tốt hơn những configuration management tool khác**.

* 1. **Một số lí do khác**
* Nguồn mở: Terraform được hỗ trợ bởi cộng đồng lớn những người đóng góp, những người xây dựng các plugin cho nền tảng này. Bất kể bạn sử dụng nhà cung cấp dịch vụ đám mây nào, thật dễ dàng tìm thấy các plugin, tiện ích mở rộng và hỗ trợ chuyên nghiệp. Điều này cũng có nghĩa là Terraform phát triển nhanh chóng, với các lợi ích và cải tiến mới được bổ sung một cách nhất quán.
* Nền tảng bất khả tri: Có nghĩa là bạn có thể sử dụng nó với bất kỳ nhà cung cấp dịch vụ đám mây nào. Hầu hết các công cụ IaC khác được thiết kế để hoạt động với một nhà cung cấp đám mây duy nhất.
* Cơ sở hạ tầng bất biến: Hầu hết các công cụ IaC đều tạo ra cơ sở hạ tầng có thể thay đổi, nghĩa là cơ sở hạ tầng có thể thay đổi để thích ứng với những thay đổi như nâng cấp phần mềm trung gian hoặc máy chủ lưu trữ mới. Mối nguy hiểm với cơ sở hạ tầng có thể thay đổi là sự trôi dạt cấu hình — khi các thay đổi chồng chất lên nhau, việc cung cấp thực tế của các máy chủ khác nhau hoặc các yếu tố cơ sở hạ tầng khác 'lệch' xa hơn so với cấu hình ban đầu, khiến lỗi hoặc các vấn đề về hiệu suất khó chẩn đoán và sửa chữa. Terraform cung cấp cơ sở hạ tầng bất biến, có nghĩa là với mỗi thay đổi đối với môi trường, cấu hình hiện tại được thay thế bằng cấu hình mới giải thích cho sự thay đổi đó và cơ sở hạ tầng được định hình lại. Thậm chí tốt hơn, các cấu hình trước đó có thể được giữ lại dưới dạng phiên bản để cho phép khôi phục nếu cần hoặc muốn.

# PHẦN 2: CÀI ĐẶT TERRAFORM

1. **TẢI TERRAFORM**

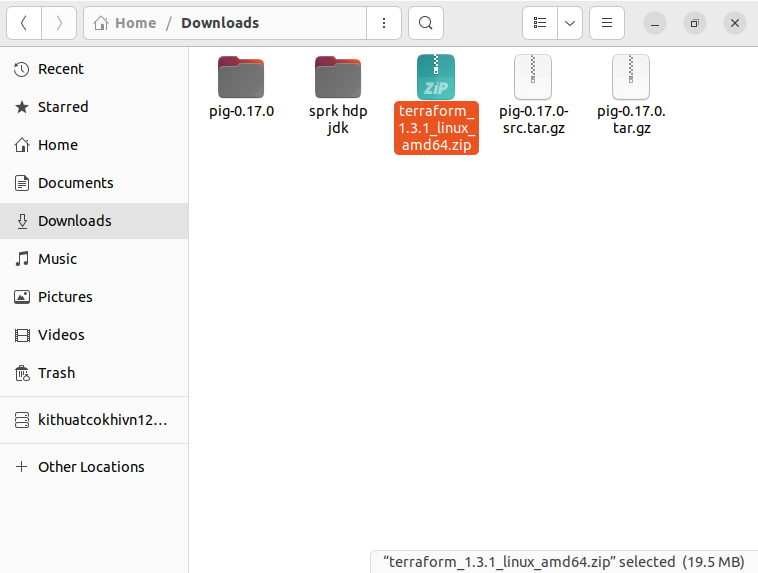
**Bước 1:** Vào trang web dowload của [Terraform](https://www.terraform.io/downloads) sau đó chọn hệ điều hành phù hợp **(cài cho cả win và ubuntu).**



Hình 4: Trang dowload Terraform

**Bước 2:**

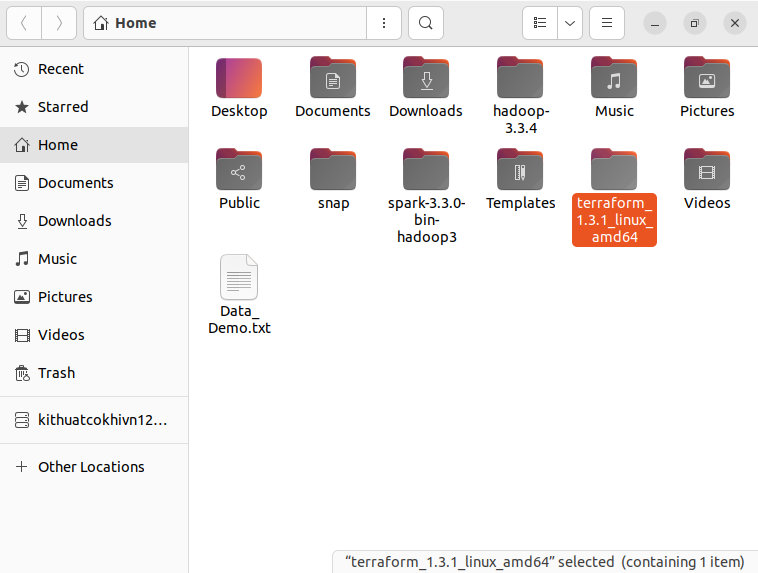
* Với Ubuntu: tải file ‘**terraform\_1.3.1\_linux\_amd64.zip**’ – mới nhất hiện tại.
* Với Win: tải file ‘**terraform\_1.3.1\_windows\_amd64.zip**’ – mới nhất hiện tại.



Hình 5: Folder cài đặt Terraform vừa tải về

**Bước 3:**

* Với Ubuntu: Giải nén folder Terraform vừa tải về sau đó di chuyển folder đến Home (bước di chuyển folder không nhất thiết phải làm).
* Với Win: Giải nén folder Terraform vừa tải về.

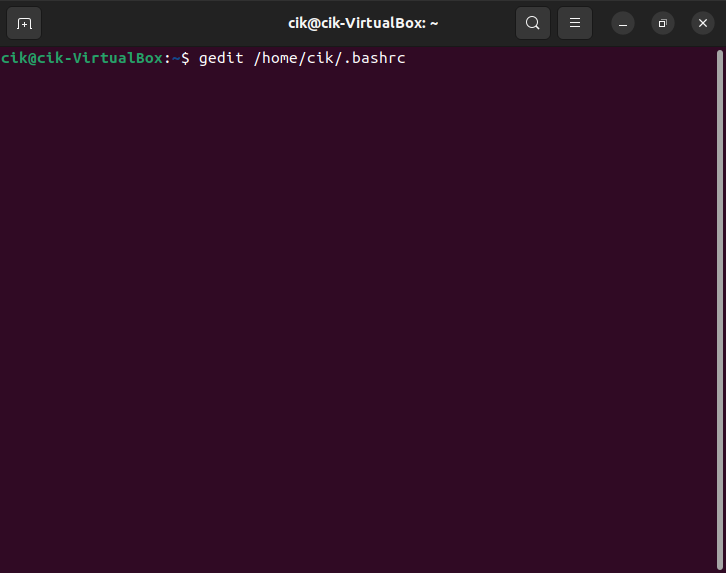


Hình 6: Folder được giải nén và chuyển đến Home

1. **CÀI ĐẶT MÔI TRƯỜNG**

Sau khi cài đặt xong thì chúng ta sẽ thiết lập môi trường cho Terraform.

* 1. **Đối với Ubuntu**

**Bước 1:** Mở Teminal và nhập “**gedit ~/.bashrc**”.

Hình 7: Câu lệnh mở file bashrc

**Bước 2:** Tạo biến sau trong file bashrc vừa mới mở:

**#Start Terra**

**export PATH=${PATH}:/home/your\_user\_name/terraform\_1.3.1\_linux\_amd64**

**#End Terra**

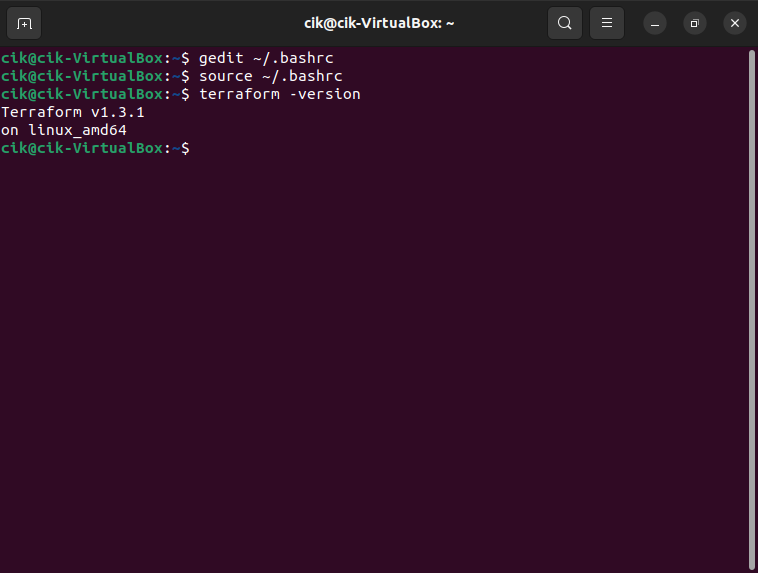
Graphical user interface, text, application

Description automatically generated

Hình 8: Tạo biến môi trường cho Terraform

**Bước 3:** Bấm save và tắt file bashrc. Sau đó nhập lệnh

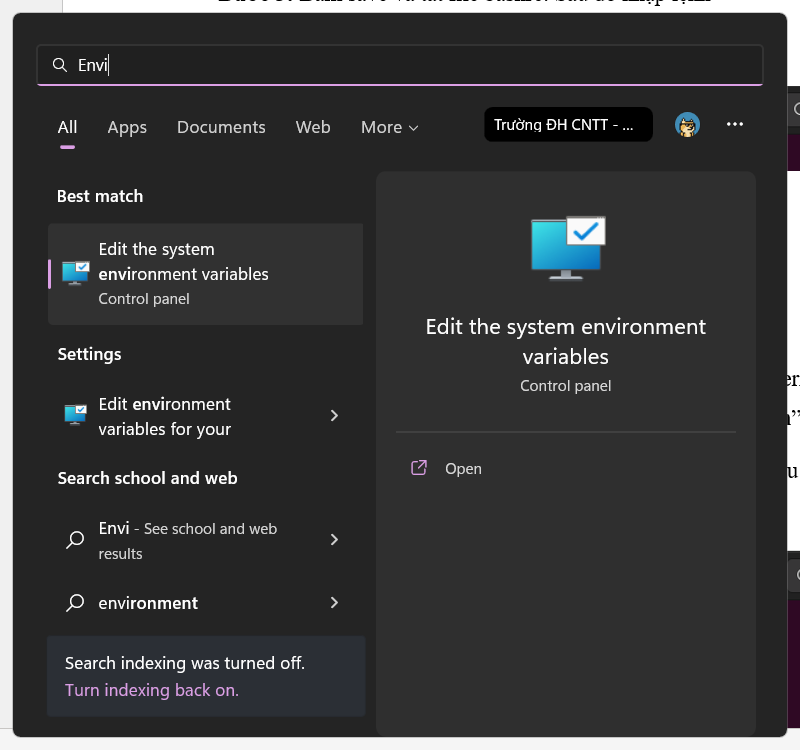
“**source ~/.bashrc**” để file đọc và thực thi file bashrc.



Hình 9: Thực thi file .bashrc

* 1. **Đối với Win**

**Bước 1:** Mở Edit the system environment variables.



Hình 10: Mở app để thay đổi môi trường

**Bước 2:** Chọn Environment Variables.

Graphical user interface, text, application, email

Description automatically generated

Hình 11: Giao diện của System Properties

**Bước 3:** Chọn Path (ở phần System varibales) => Chọn New => Thêm đường

dẫn folder Terraform vừa mới giải nén => Lưu.

Graphical user interface, text, application

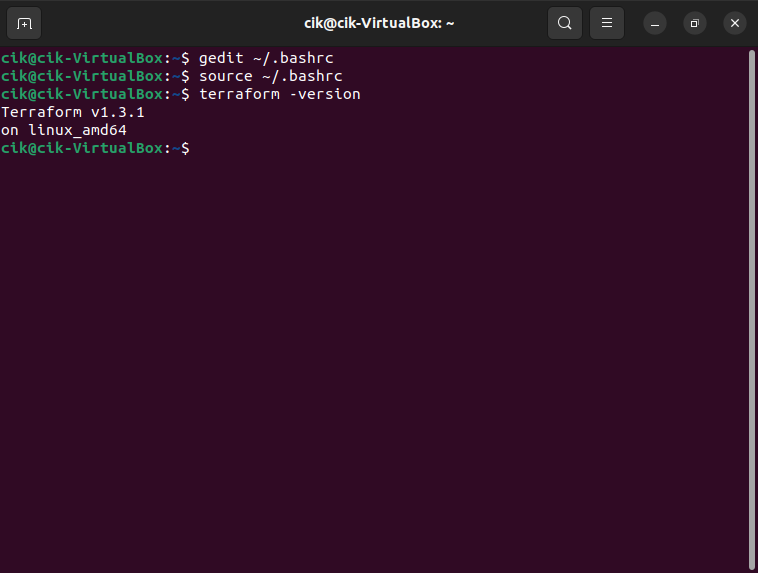
Description automatically generated

Hình 12: Thêm đường dẫn Terra vào môi trường

1. **KIỂM TRA CÀI ĐẶT**

Sau khi đã làm các bước trên cuối cùng chúng ta sẽ kiểm tra Terraform đã được cài thành công hay chưa bằng cách nhập lệnh “**terraform -version**”.

Nếu cài thành công thì sẽ hiện lên phiên bản của Terraform, nếu không thì hãy xem xét lại các bước trên.



Hình 13: Kiểm tra cài đặt Terraform

# PHẦN 3: DEMO

# PHẦN 4: TÀI LIỆU THAM KHẢO

[1] Viblo, “Infrastructure as Code là gì ? tại sao lại chọn Terraform ? - Website chia sẻ kiến thức,” [https://viblo.asia/p/infrastructure-as-code-la-gi-tai-sao-lai-chon-terraform-LzD5dwrEljY](https://viblo.asia/p/infrastructure-as-code-la-gi-tai-sao-lai-chon-terraform-LzD5dwrEljY%20) (truy cập lần cuối 29/09/2022).

[2] Whizlabs, “Terraform vs CloudFormation - Website chia sẻ kiến thức[,” https://www.whizlabs.com/blog/terraform-vs-cloudformation-vs-ansible/](https://uithcm-my.sharepoint.com/personal/19521694_ms_uit_edu_vn/Documents/,) (truy cập lần cuối 29/09/2022).

[3] TechTarget, “DEFINITION Terraform - Website chia sẻ kiến thức” [https://www.techtarget.com/searchitoperations/definition/Terraform](https://www.techtarget.com/searchitoperations/definition/Terraform%20)

(truy cập lần cuối 29/09/2022).

[4] HashiCorpTerraform, “What is Terraform? - Website chính của Terraform” [https://www.terraform.io/intro](https://www.terraform.io/intro%20) (truy cập lần cuối 29/09/2022).