Ansible vs Terraform vs Puppet 차이점 및 선택

Rainbound-IT

'DevOps' 세계에서 조직은 IAC(Infrastructure as Code)를 사용하여 프로세스를 구현하거나 구축하고 있습니다. Ansible, Terraform 및 Puppet을 통해 기업은 올바른 결과를 지속적으로 보장하기 위해 절차를 테스트하고 시행하는 반복 가능한 구성을 확장하고 생성할 수 있습니다.

이 세 가지의 차이점을 더 자세히 살펴보겠습니다. 귀하의 요구 사항에 가장 적합한 플랫폼을 선택하도록 안내합니다. 세 가지 모두 매우 복잡한 요구 사항이 있는 복제 가능하고 반복적인 애플리케이션을 배포하기 위한 고급 수준 플랫폼입니다.

구성 관리, 아키텍처 및 오케스트레이션 측면에서 이러한 애플리케이션의 유사점과 차이점을 비교하고 정보에 입각한 결정을 내립니다.

코드로서의 인프라

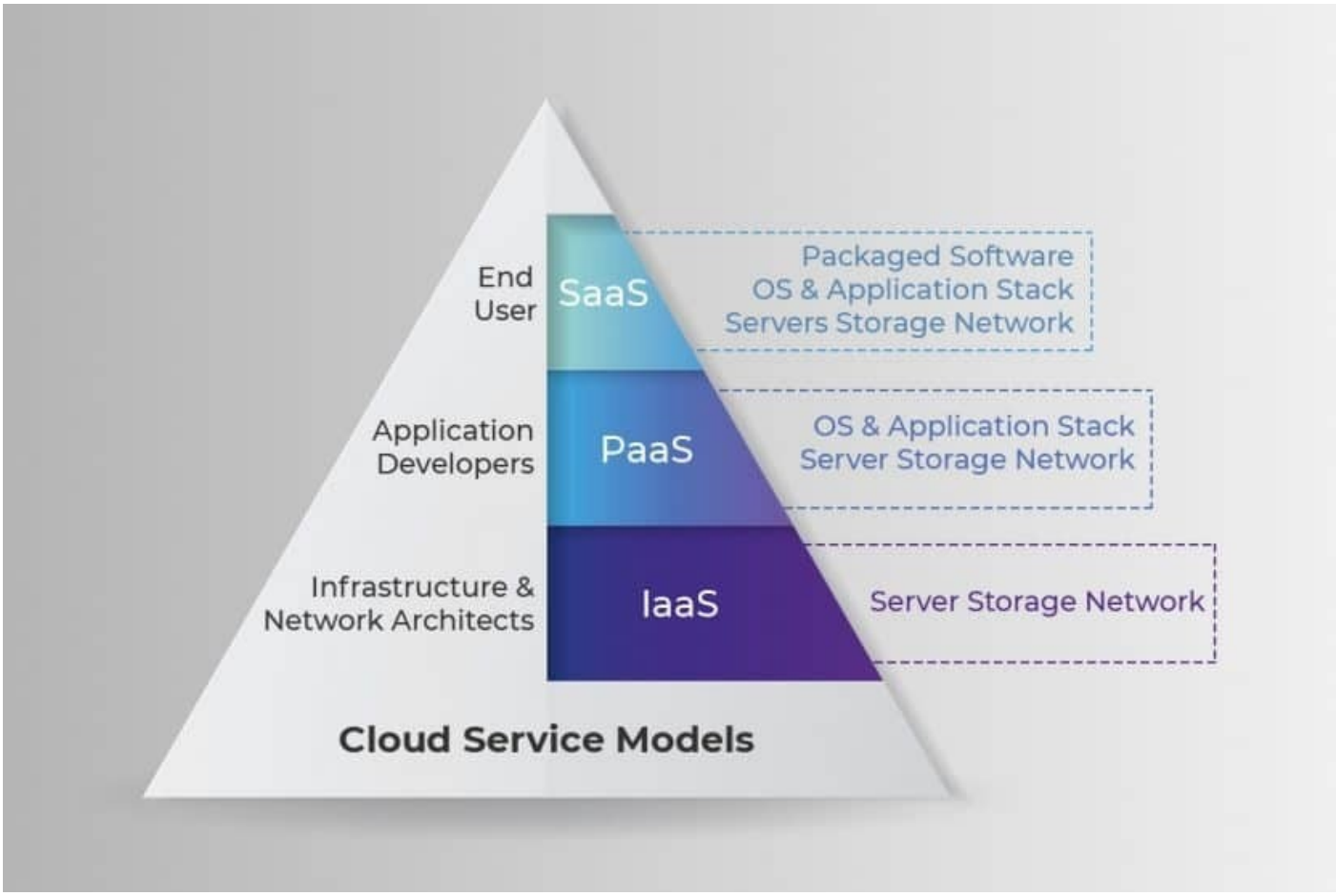
10여 년 전에 도입된 IAC(Infrastructure as Code) 개념은 컴퓨터 데이터 센터를 관리하고 프로비저닝하는 프로세스를 나타냅니다. 데이터 센터 서버, 네트워킹 인프라 및 스토리지를 관리하기 위한 전략입니다. 그 목적은 대규모 관리 및 구성을 극적으로 단순화하는 것입니다.

IAC를 사용하면 도구나 물리적 하드웨어를 구성하지 않고도 기계가 읽을 수 있는 정의 파일을 통해

컴퓨터 데이터 센터를 프로비저닝하고 관리할 수 있습니다. 간단히 말해서 IAC는 수동 구성, 빌드 가이드, 실행 책 및 관련 절차를 코드로 취급합니다. 인프라 상태를 유지 관리하는 코드인 소프트웨어로 읽습니다.

구성 드리프트, 시스템 불일치, 인적 오류 및 컨텍스트 손실을 해결하도록 설계된 IAC는 잠재적으로 심각한 문제를 해결합니다. 이러한 프로세스에는 상당한 시간이 소요되었습니다. 최신 IAC 도구는 모든 프로세스를 더 빠르게 만듭니다. 수동 구성 단계를 없애고 반복 가능하고 확장 가능하게 만듭니다. 수백 대의 서버를 훨씬 더 빠르게 로드할 수 있습니다. 이를 통해 사용자는 예측 가능한 아키텍처를 확보하고 데이터 센터의 구성 또는 상태를 자신 있게 유지할 수 있습니다.

Ansible, Terraform 및 Puppet의 세 가지 주요 예와 함께 선택할 수 있는 여러 IAC 도구가 있습니다. 그들 모두는 고유한 강점과 약점을 가지고 있으며, 이에 대해서는 더 자세히 살펴보겠습니다.



Terraform, Ansible 및 Puppet에 대한 짧은 배경 지식

도구 비교를 시작하기 전에 아래의 간단한 설명을 참조하십시오.

Terraform(2014년 출시 – 현재 버전 0.12.8): Hashicorp 는 인프라 조정자 및 서비스 제공자로 Terraform 을 개발했습니다 . 클라우드에 구애받지 않고 여러 공급자를 지원합니다. 결과적으로 사용자는 동일한 프로그래밍 언어 및 구성 구성을 사용하여 다중 클라우드 또는 다중 오퍼링 환경을 관리할 수 있습니다. Haschorp Language를 사용하며 다른 도구에 비해 상당히 사용자 친화적입니다.

Ansible(2012년 출시 – 현재 버전 2.8.4): Ansible 은 다양한 클래스와 구성 방법을 활용하여 서

비스와 서버를 원하는 상태로 만드는 데 사용되는 강력한 도구입니다. 또한 래퍼 모듈을 통해 다른 공급자에 연결하여 리소스를 구성할 수도 있습니다. 사용자는 빠른 배포 기능과 함께 코딩과 관련하여 가벼우므로 선호합니다.

Puppet(2005년 출시 – 현재 버전 6.8.0): Puppet 은 사용 가능한 가장 오래된 선언적 원하는 상태 도구 중 하나입니다. 카탈로그를 통해 클라이언트의 상태를 새로 고치는 서버/클라이언트 기반입니다. 강력한 메타데이터 구성 방법인 "hiera data"를 사용합니다. 인프라를 코드로 정의하는 능력을 통해 프로그램으로 시스템 구성을 시행합니다. 여러 애플리케이션 서버에서 동시에 문자열을 가져오기 위해 Windows 또는 Linux에서 널리 사용됩니다.

오케스트레이션 대 구성 관리

Ansible과 Terraform에는 몇 가지 중요한 차이점이 있지만 둘은 몇 가지 유사점도 있습니다. 도구 유형인 오케스트레이션과 구성 관리라는 두 가지 DevOps 개념을 보면 다릅니다. Terraform은 오케스트레이션 도구입니다. Ansible은 주로 CM(구성 관리 도구)입니다. 그들은 다르게 수행되지만 이러한 기능은 상호 배타적이지 않기 때문에 일부 중복이 있습니다. 다양한 용도와 장점에 최적화되어 상황에 맞게 사용하세요.

오케스트레이션 도구 에는 한 가지 목표가 있습니다. 즉, 환경이 지속적으로 '원하는 상태'에 있도록 하는 것입니다. Terraform은 환경의 상태를 저장하기 때문에 이를 위해 구축되었으며, 무언가가 제대로 작동하지 않을 경우 다시 로드한 후 시스템을 자동으로 계산하고 복원합니다. 일정하고 변하지 않는 상태가 필요한 환경에 적합합니다. 'Terraform Apply'는 모든 이상 현상을 효율적으로 해결하기 위해 만들어졌습니다.

구성 관리 도구는 다릅니다. 그들은 시스템을 재설정하지 않습니다. 대신 로컬에서 문제를 해결합니다. Puppet은 서버에 소프트웨어를 설치하고 관리하는 디자인을 가지고 있습니다. Puppet과 마찬가지로 Ansible도 각 작업과 도구를 구성하고 오류나 손상 없이 올바르게 작동하는지 확인할 수 있습니다. CM 도구는 시스템을 완전히 교체하는 대신 문제를 해결하기 위해 작동합니다. 이 경우 Ansible은 오케스트레이션을 수행하고 인프라를 교체할 수 있기 때문에 약간의 하이브리드입니다. Terraform이 더 널리 사용됩니다. Ansible에는 없는 고급 상태 관리 기능을 가지고 있기 때문에 우수한 제품으로 간주됩니다.

여기서 알아야 할 중요한 점은 여기에 기능이 중복된다는 것입니다. 대부분의 CM 도구는 일정 수준의 프로비저닝을 수행할 수 있으며 그 반대의 경우 많은 프로비저닝 도구가 약간의 구성 관리를 수행할 수 있습니다. 현실은 다양한 도구가 특정 유형의 작업에 더 적합하므로 서버 요구 사항에 따라 달라집니다.

절차적 vs. 선언적

DevOps 도구 는 작업을 정의하는 '선언적' 및 '절차적'이라는 두 가지 범주로 제공됩니다. 겹침이 존재하기 때문에 모든 도구가 이 금형에 맞는 것은 아닙니다. 절차적은 코드에 배치해야 하는 정확한 방향과 절차가 필요한 도구를 정의합니다. 선언적이란 필요한 것을 정확히 '선언'하는 도구를 말합니다. 결과를 얻는 데 필요한 프로세스를 설명하지 않습니다.

이 경우 Terraform은 완전히 선언적입니다. 정의된 환경이 있습니다. 해당 환경에 변경 사항이 있으면 다음 'Terraform Apply'에서 수정됩니다. 간단히 말해서 이 도구는 시스템 관리자가 설명한 대로 원하는 종료 상태에 도달하려고 시도합니다. Puppet은 또한 이러한 방식으로 선언적인 것을 목표로 합니다.
Terraform을 사용하면 원하는 상태를 설명하기만 하면 Terraform이 자동으로 한 상태에서 다음 상태로 이동하는 방법을 알아낼 것입니다.

이 경우 Ansible은 일종의 하이브리드입니다. 그것은 둘 다 약간 할 수 있습니다. 절차적 스타일 구성을 구현하고 선언적 스타일로 수행되는 대부분의 모듈을 사용하는 임시 명령을 수행합니다.

Ansible을 사용하기로 결정했다면 설명서를 주의 깊게 읽어서 Ansible의 역할과 예상되는 동작을 이해하십시오. 올바른 결과를 얻기 위해 리소스를 추가하거나 빼야 하는지 또는 명시적으로 필요한 리소스를 표시해야 하는지 여부를 알아야 합니다.

프로비저닝 비교

모든 인프라 프로비저닝을 자동화하는 것은 애플리케이션 및 배포의 전체 운영 수명 주기를 자동화하는 첫 번째 단계입니다. 클라우드에서 소프트웨어는 VM, Docker 컨테이너 또는 베어메탈 서버 에서 실행 됩니다. Terraform 또는 Ansible은 이러한 시스템을 프로비저닝하는 데 좋은 선택입니다. Puppet은 이전 도구이므로 여러 서버를 관리하기 위한 최신 DevOps 프로그램을 자세히 살펴보겠습니다.

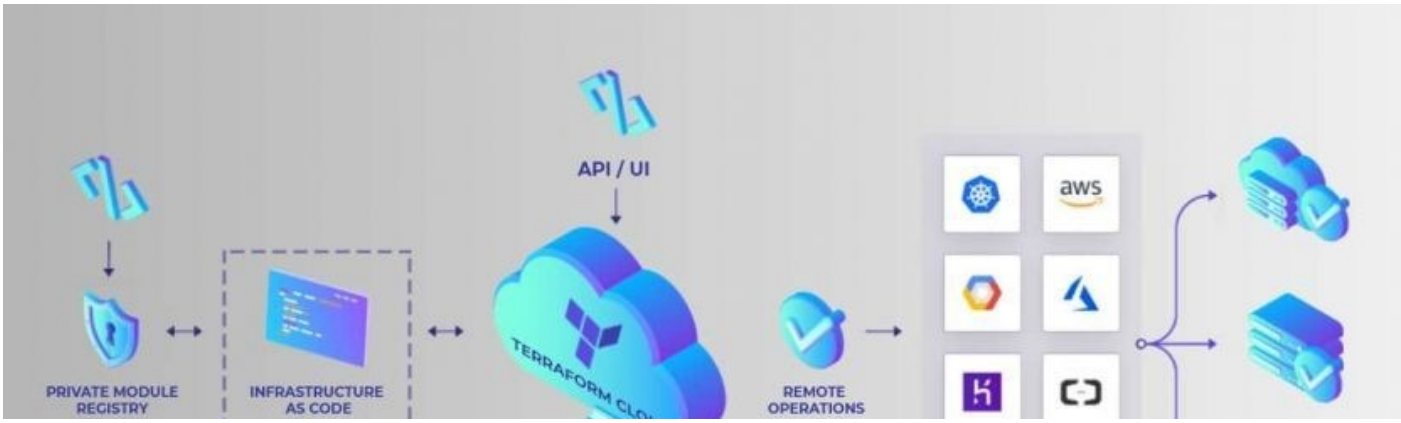
Terraform과 Ansible은 아래에 설명된 것처럼 프로비저닝 프로세스에 다르게 접근하지만 일부 겹치는 부분이 있습니다.

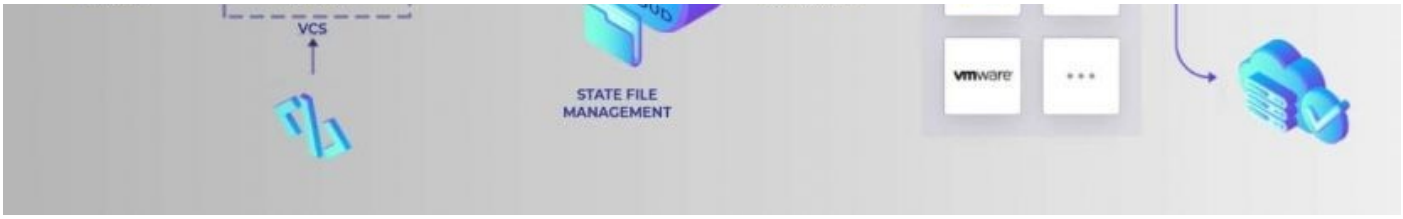
Terraform으로 프로비저닝:

Terraform의 기존 선언적 모델에는 표시되지 않는 특정 동작이 있습니다. 이 설정은 다음과 같은 방식으로 Terraform을 사용할 때 상당한 양의 불확실성과 복잡성을 추가합니다.

Terraform 모델은 계획의 일부일 때 제공자의 작업을 모델링할 수 없습니다. 프로비저닝 도구를 성공적으로 사용하려면 일반적인 Terraform 사용에 필요한 것보다 더 많은 세부 사항을 조정해야 합니다.

사용자 서버에 대한 직접 네트워크 액세스 권한 부여, 필수 외부 소프트웨어 설치, 로그인을 위한 Terraform 자격 증명 발급과 같은 추가 조치가 필요합니다.



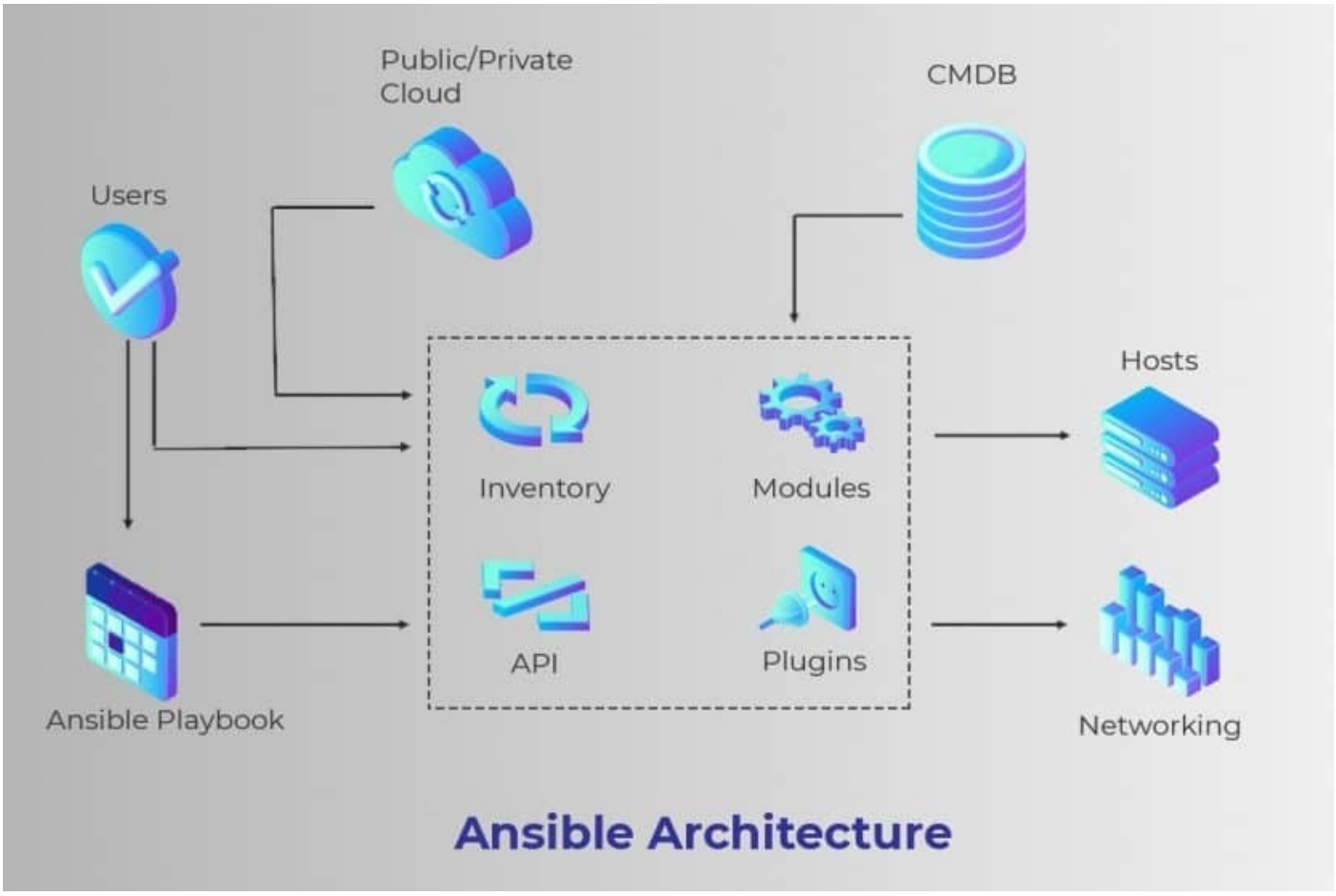


Ansible로 프로비저닝:

Ansible은 최신 클라우드 플랫폼, 네트워크 장치, 베어메탈 서버, 가상화된 호스트 및 하이퍼바이저를 안정적으로 프로비저닝할 수 있습니다.

부트스트랩을 완료한 후 Ansible은 별도의 팀이 노드를 스토리지에 연결할 수 있도록 합니다. 로드 밸런서, 보안 패치 또는 기타 운영 작업에 추가할 수 있습니다. 이 설정은 Ansible을 모든 프로세스 파이프라인에 대한 완벽한 연결 도구로 만듭니다.

이는 일상적인 관리를 통해 베어 인프라를 자동으로 가져오는 데 도움이 됩니다. Ansible을 통한 프로비저닝을 통해 사용자는 구성 관리, 애플리케이션 배포 및 오케스트레이션 전반에 걸쳐 사람이 읽을 수 있는 보편적인 자동화 언어를 원활하게 사용할 수 있습니다.



AWS용 Ansible과 Terraform의 차이점

AWS는 Amazon의 자회사인 Amazon Web Services의 약자로 개인, 회사 및 기업체에 주문형 클라우드 컴퓨팅 플랫폼을 제공합니다. Terraform과 Ansible은 모두 AWS 관리를 상당히 다르게 취급합니다.

AWS를 통한 Terraform:

Terraform은 가상화 경험이 많지 않은 사용자가 AWS를 관리할 수 있는 탁월한 방법입니다. 처음에는 상당히 복잡하게 느껴질 수 있지만 Terraform은 채택을 늘리는 데 방해가 되는 장애물을 크게 줄였습니다.

AWS와 함께 Terraform을 사용할 때 몇 가지 주목할만한 이점이 있습니다.

Terraform은 오픈 소스로, 오픈 소스 소프트웨어를 사용하는 일반적인 이점과 그 뒤에 있는 사용자 커뮤니티가 증가하고 있습니다.

리소스 관계에 대한 기본적 이해가 있습니다.

장애가 발생하면 종속 리소스로 격리됩니다. 반면에 비의존적 리소스는 계속 생성, 업데이트 및 소멸됩니다.

Terraform은 사용자에게 변경 사항을 적용하기 전에 미리 볼 수 있는 기능을 제공합니다.

Terraform은 JSON 지원 및 사용자 친화적인 사용자 정의 구문과 함께 제공됩니다.

AWS로 앤서블:

Ansible은 오랫동안 AWS에 대한 상당한 지원을 제공해 왔습니다. 이 지원을 통해 Ansible 플레이북을 사용하여 가장 복잡한 AWS 환경도 해석할 수 있습니다. 일단 설명되면 사용자는 다양한 지역에서 수백, 수천 개의 인스턴스로 확장할 수 있는 기능을 사용하여 필요에 따라 여러 번 배포할 수 있습니다.

Ansible에는 AWS 기능을 지원하는 100개에 가까운 모듈이 있습니다. Virtual Private Cloud(VPC), Simple Storage Service(S3), Security Token Service, Security Groups, Route53, Relational Database Service, Lambda, Identity Access Manager(IAM), AMI 관리, CloudTrail 등. 또한 사용자의 Linux, Windows, UNIX 등의 다양한 측면을 관리하기 위한 1300개 이상의 추가 모듈이 포함되어 있습니다.

다음은 AWS와 함께 Ansible을 사용할 때의 이점입니다.

Ansible Tower의 클라우드 인벤토리 동기화를 사용하면 시작 방법에 관계없이 어떤 AWS 인스턴스가 등록되는지 정확하게 알 수 있습니다.

수명 주기를 통해 배포된 인프라를 정확하게 추적하여 인벤토리를 제어할 수 있습니다. 따라서 시스템이 제대로 관리되고 보안 정책이 올바르게 실행되는지 확인할 수 있습니다.

사용자가 작업을 수행하는 데 필요한 AWS 리소스에만 액세스할 수 있도록 하는 역할 기반 액세스 제어 세트를 통한 자동화의 안전성.

동일한 간단한 플레이북 언어가 인프라를 관리하고 애플리케이션을 대규모로 다른 인프라에 쉽게 배포합니다.

Ansible, Puppet 및 Terraform의 비교

Puppet, Terraform 및 Ansible은 상당한 기간 동안 사용되었습니다. 그러나 설정, GUI, CLI, 언어, 사용법 및 기타 기능에 관해서는 다릅니다.

	Source	Cloud	Type	Infrastructure	Language	Agent	Master	Community	Maturity
Puppet	Open	All	Config Mgmt	Mutable	Declarative	Yes	Yes	Large	High
Ansible	Open	All	Config Mgmt	Mutable	Procedural	No	No	Huge	Medium
Terraform	Open	All	Config Mgmt	Mutable	Declarative	No	No	Huge	Low

아래 세 가지를 자세히 비교할 수 있습니다.

차이점	**Ansible**	**Puppet**	**Terraform**
관리 및 일정	Ansible에서는 서버가 구성을 노드에 푸시하기 때문에 즉각적인 배포가 가능합니다. 일정 관리와 관련하여 엔터프라이즈 버전인 Ansible Tower에는 기능이 있지만 무료 버전에는 없는 기능이 있습니다.	Puppet은 클라이언트가 서버에서 구성을 가져오는 푸시 및 풀 구성에 주로 중점을 둡니다. 구성은 Puppet의 언어로 작성해야 합니다. 예약과 관련하여 Puppet의 기본 설정을 통해 모든 노드가 원하는 상태인지 확인할 수 있습니다.	Terraform에서 리소스 스케줄러는 제공자와 유사하게 작동하여 리소스를 요청할 수 있습니다. 따라서 AWS와 같은 물리적 공급자에 국한되지 않고 계층에서 사용할 수 있습니다. Terraform을 사용하여 스케줄러를 실행하는 물리적 인프라를 설정하고 예약된 그리드에 프로비저닝할 수 있습니다.
간편한 설정 및 사용	Ansible은 설치 및 사용이 더 간단합니다. 클라이언트 시스템에서 실행되는 에이전트가 없는 마스터가 있습니다. 에이전트가 없다는 사실은 단순성에 크게 기여합니다. Ansible은 대부분의 Linux 및 Unix 배포에 기본 제공되는 Python 언어로 작성된 YAML 구문을 사용합니다.	Puppet은 시스템 관리자를 위한 모델 기반입니다. Puppet 서버는 하나 이상의 서버에 설치할 수 있지만 Puppet Agent는 관리가 필요한 모든 노드에 설치해야 합니다. 따라서 모델은 클라이언트-서버 또는 에이전트-마스터 모델입니다. 설치 시간은 약 10분에서 30분 정도 소요될 수 있습니다.	Terraform은 설정 및 사용법에 관해서도 이해하기 쉽습니다. 설치 프로그램을 실행하는 데 필요한 경우 사용자가 프록시 서버를 사용할 수도 있습니다.
유효성:	Ansible에는 활성 노드가 떨어질 경우에 대비하여 보조 노드가 있습니다.	Puppet에는 원래 마스터가 실패할 경우를 대비하여 하나 이상의 마스터가 있습니다.	Terraform의 경우에는 해당되지 않습니다.
확장성:	확장성을 달성하기 더 쉽습니다.	확장성은 달성하기가 쉽지 않음	확장성이 비교적 쉽게 달성됨

모듈	Ansible의 저장소 또는 라이브러리를 Ansible Galaxy라고 합니다. 별도의 분류 기능이 없으며 수동 개입이 필요합니다.	Puppet의 저장소 또는 라이브러리를 Puppet Forge라고 합니다. 6000개에 가까운 모듈이 포함되어 있습니다. 사용자는 Puppet 모듈을 Puppet이 승인하거나 지원하는 것으로 표시하여 상당한 시간을 절약할 수 있습니다.	Terraform의 경우 모듈을 통해 사용자는 재사용 가능한 부분을 추상화할 수 있습니다. 이러한 부분은 한 번 구성할 수 있으며 모든 곳에서 사용할 수 있습니다. 따라서 사용자는 리소스를 그룹화하고 입력 및 출력 변수를 정의할 수 있습니다.
GUI	덜 개발된 Ansible의 GUI는 처음에 명령줄 전용 도구로 도입되었습니다. 엔터프라이즈 버전은 UI를 제공하지만 명령줄과의 동기화 문제로 인해 여전히 기대에 미치지 못합니다.	Puppet의 GUI는 많은 복잡한 작업을 수행할 수 있는 Ansible의 GUI보다 우수합니다. 활동을 효율적으로 관리, 보기 및 모니터링하는 데 사용됩니다.	Terraform에는 타사 GUI만 사용할 수 있습니다. 예를 들어, Codeherent의 Terraform GUI입니다.
지원하다	Ansible에는 엔터프라이즈 버전에 대한 두 가지 수준의 전문 지원도 포함되어 있습니다. 또한, 사용자와 기여자가 큰 모임인 AnsibleFest가 매년 개최됩니다. Puppet에 비해 커뮤니티는 작습니다.	Puppet에는 지식 기반과 함께 전용 지원 포털이 있습니다. 또한 두 가지 수준의 전문 지원이 있습니다. 스탠다드와 프리미엄. "DevOps 상태" 보고서는 Puppet 커뮤니티에서 매년 생성합니다.	Terraform은 웹 포털을 통해 HashiCorp의 지원 채널에 직접 액세스할 수 있습니다.

고려해야 할 세 가지 종합적인 솔루션

위의 비교를 살펴본 후 Ansible은 다른 것에 비해 스크립트와 같은 방식으로 시스템을 저장하고 구성하는 데 매우 유용합니다. 사용자는 수명이 짧은 환경에서 효율적으로 작업할 수 있습니다. 또한 컨테이너 호스트를 구성하기 위해 Kubernetes와 원활하게 작동합니다.

Puppet은 커뮤니티 지원과 관련하여 더 성숙합니다. Puppet에는 엔터프라이즈급 솔루션으로 더 많이 작동할 수 있는 우수한 모듈이 있습니다. 강력한 모듈 테스트는 사용하기 쉽습니다. Ansible은 소규모, 임시 및 빠른 배포에 적합합니다. 반면 Puppet은 장기 또는 더 복잡한 배포에 권장되며 Docker 컨테이너 및 컨테이너 오케스트레이터 를 관리할 수 있습니다 .

Terraform은 서버 아래의 클라우드 서비스를 관리할 때 더 나은 성능을 보입니다. Ansible은 소프트웨어 및 머신 프로비저닝에 탁월합니다. Terraform은 클라우드 리소스 관리에 탁월합니다.

세 가지 모두 자동화를 위한 IAC 환경을 설계할 때 장단점이 있습니다. 성공은 어떤 작업에 어떤 도구를 사용할지 아는 데 달려 있습니다.

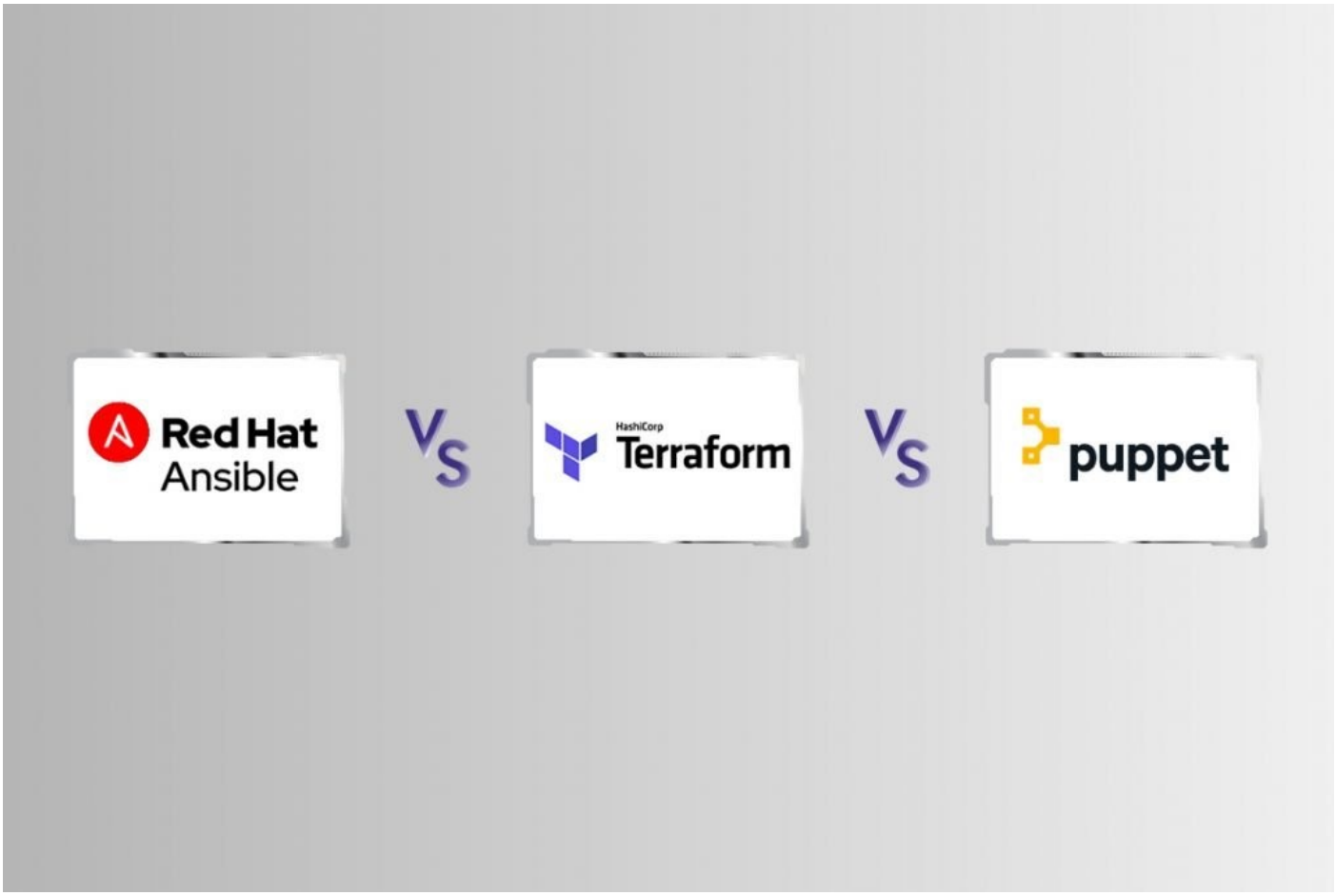
Reference

https://phoenixnap.com/blog/ansible-vs-terraform-vs-puppet

Ansible vs Terraform vs Puppet: Which to Choose?

We compare the three most used IAC tools on the market, Ansible, Terraform, and Puppet. Find out more about their features, advantages, and drawbacks today.

phoenixnap.com



Chef, Puppet, Ansible, SaltStack 또는 CloudFormation이 아닌 Terraform을 사용하는 이유

https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c

Why we use Terraform and not Chef, Puppet, Ansible, SaltStack, or CloudFormation

Update, November 17, 2016: We took this blog post series, expanded it, and turned it into a book called Terraform: Up & Running!

blog.gruntwork.io