# 온프레미스 환경에서의 코드형 인프라 기반 쿠버네티스 구축 옵션 비교 분석

현혜연[1], 김가현[1], 김미진[1], 이지연[2], 유지원[3], 김성민[4]
성신여자대학교 융합보안공학과[1], 컴퓨터공학과[2], 학부생
성신여자대학교 융합보안공학과 석사과정[3], 교수[4]
{20221143, 20211039, 20221081, 20201006, 220246055, sm.kim}@sungshin.ac.kr

# Comparative Analysis of Infrastructure as Code Options for Building Kubernetes in On-Premise Environments

Hye-Yeon Hyeon[1], Ka-Hyun Kim[1], Mi-Jin Kim[1], Ji-Yeon Lee[2], Ji-Won Yoo[2], Seng-Min Kim[1]
[1]Dept. of Convergence Security Engineering, Sungshin Women's University
[2]Dept. of Computer Engineering, Sungshin Women's University

요 약

최근 클라우드 인프라 구축에서 보안까지 통합 관리하는 DevSecOps의 중요성이 커지고 있으며, 코드형 인프라는 이를 자동화하는 핵심 기술로 활용되고 있다. 특히 Terraform이 가장 많이 사용되고 있는데, 이것은 Public cloud뿐 아니라 On-premise 환경에서도 Kubernetes를 자동으로 구성할 수 있는 도구이다. 본 연구는 On-premise 환경에서 Terraform을 활용해 Kubernetes를 구축할 수 있는 세 가지 Matchbox, Rancher, Proxmox 기반 방식을 비교 분석하였다. 이를 통해 On-premise 환경에서도 Terraform 기반의 IaC를 활용한 Kubernetes 구축이 충분히 실현 가능함을 입증하였다.

## 1. 서론

최근 클라우드 환경을 구축할 때 인프라 설정부터 보안까지 한 번에 관리하는 DevSecOps가 중요하게 고려된다. DevSecOps 내 CI/CD(Continuous Integration/ Continuous Deployment) 파이프라인에서 핵심 기반 기술 중 하나는 인프라를 코드로 관리하는 코드형 인프라(Infrastructure as Code, IaC)를 통한 자동화이다. IaC를 사용하여 코드로 클라우드 인프라를 배포, 관리, 프로비저닝할 수 있으며 그 중 Terraform이 많이 사용되고 있다[1].

클라우드 네이티브 기술의 발전에 따라, 클라우드 인프라의 운영 형태가 퍼블릭 클라우드 중심에서 프라이빗 클라우드 및 하이브리드 클라우드로 다변화되고 있으며, DevSecOps의 적용 또한 이러한 흐름과 함께 확장되고 있다. 현재 IaC의 활용은 온프레미스보다 퍼블릭 클라우드에 초점이 맞춰져 있으며, 온프레미스에서 쿠버네티스를 구축할 때 Terraform을 활용한 실제 적용 사례나 활용 경험에 대한 공유는 아직 충분하지 않다. 구체적으로 Matchbox, Rancher, Proxmox 등을 Terraform provider로 지정하여 온프레미스에서 쿠버네티스를 구축하는 것이 가능하나, 어떠한 방법이 상황에 따라 효과적인지에 대한 논의는 부재하다.

본 연구에서는 프라이빗 또는 하이브리드 클라우드 환경에서의 쿠버네티스 클러스터를 구축할 수 있는 다양한 옵션들을 조사하고, 비교 분석을 진행하였다. 이를 통해, On-premise 환경에서의 코드형 인프라 기반 Kubernetes 구축 시 인프라 운영자의 고려 사항에 따라 어떠한 공급자를 baseline으로 하는 것이 적절한지 교훈을 제시한다.

## 2. 비교 분석

Matchbox, Rancher, Proxmox를 각각 provider로 사용하여 동일한 Kubernetes 클러스터를 구성하여 Master node 1개와 Worker node 2개를 구축하였다.

**Matchbox 기반 구성**: PXE(Pre-boot eXecution Environment) 부팅을 전제로 하며, libvirt provider를 사용해 PXE 부팅이 가능한 VM 3대를 생성한다. API 엔드포인트와 클라이언트 인증서를 기반으로 provider가 구성되며, 각 노드에서 PXE 요청을 수신

하면 MAC 주소를 기준으로 ignition config를 전달해 Fedora CoreOS를 자동 설치한다. 이때 kubelet 설정, 마스터·워커 역할 구분, SSH 공개키 등록 등이 포함되어 설치 후 기본적인 환경설정을 해준다. Typhoon 모듈은 이러한 ignition 설정과 Kubernetes 클러스터 구성을 Terraform으로 통합 관리하며, 마스터 및 워커 노드의 구성과 쿠버네티스 설치를 자동화한다.

**Rancher 기반 구성**: libvirt_domain.k8s_node 리소스를 반복 설정(count = 3)하여 3개의 VM을 생성하며, 각 VM은 로컬 서버에 미리 다운로드한 Ubuntu 20.04 LTS 기반 qcow2 이미지를 사용한다. 초기 설정은 cloud-init을 통해 SSH 접속, Docker 설치 등을 자동화하여 수행한다. 이후 RKEE(Rancher Kubernetes Engine provider)를 이용해 각 VM에 SSH로 접속하고, Kubernetes 클러스터에 필요한 구성요소를 역할에 따라 설치한다. RKE는 SSH 접근과 Docker가 설치된 환경을 전제로 하며, locals.node_ips를 통해 각 VM의 IP를 추출해 배열로 구성하고 이를 RKE에 전달함으로써 노드 설정을 자동화한다.

**Proxmox 기반 구성:** PVE(Proxmox Virtual Environment)의 cloud-init 지원 VM 템플릿을 활용하여 가상 머신 환경을 구성할 수 있으며, Terraform과의 연동을 통해 노드 생성 및 초기 설정의 자동화가 가능하다. Terraform은 telmate/proxmox 프로바이더를 통해 Proxmox API에 접근하고, proxmox_vm_qemu 리소스를 이용하여 지정된 템플릿 기반의 가상 머신을 생성한다. 이 과정에서 Terraform은 SSH 공개키, 사용자 계정, 호스트명, 네트워크 구성 등의 초기 설정 정보를 cloud-init을 통해 전달하며, 해당 설정은 가상 머신 부팅 시 자동으로 적용된다. Terraform 코드는 provier.tf, credentials.auto.tfvars, vm-qemu-kube.tf, kube-cluster-config.auto.tfvars 파일로 구성된다. provider.tf은 proxmox와 terraform 상호작용을 위해 proxmox provider을 지정하는 파일이다. credentials.auto.tfvars은 provider 을 제외한 변수 저장 파일로 민감한 정보를 저장한다. kube-cluster-config.auto.tfvars은 노드별로 다른 vmid, 이름, MAC 주소 등의 값을 변수로 정의하면, Terraform으로 여러 노드를 생성할 때 설정값 충돌 없이 일관성을 유지할 수 있다. vm-qemu-kube.tf는 QEMU VM 생성을 위한 Terraform 구성 파일이며, kube-cluster-config.auto.tfvars에 정의된 값을 기반으로 여러 VM을 자동으로 생성하고 설정한다. 웹 기반 GUI로 VM 상태 확인, 리소스 모니터링, 콘솔 접속 등을 지원한다. Terraform 기반 자동화와 GUI도 함께 사용할 수 있어 유연한 인프라 관리가 가능하다.

## 3. 결론

세 가지 방식 각각이 요구하는 구성 요소와 복잡도, 설정 방식의 차이를 확인하였다.

Matchbox 기반 구성은 PXE 부팅과 Ignition을 통해 Fedora CoreOS를 자동 설치하고, 초기 구성 전반을 완전히 자동화할 수 있어 대규모 베어메탈 환경에 용이하다. 다만, 초기 PXE 환경 구성과 Ignition 파일 설계가 복잡하여 높은 수준의 사전 지식과 준비가 요구된다. Rancher 기반 구성은 libvirt를 활용한 VM 생성과 cloud-init 기반 초기 자동화를 통해 중소규모의 개발 및 테스트 환경에 효율적인 대안으로 활용될 수 있다. 자동화 범위는 상대적으로 제한적이므로 대규모 환경에는 다소 부적합할 수 있다. Proxmox 기반 구성은 자체 가상화 인프라인 Proxmox를 이미 활용 중인 조직에 적합하며, 웹 기반 GUI와 Terraform을 병행하여 VM 인프라를 유연하게 구성할 수 있다. cloud-init로 자동화 설정이 용이하지만 효과적인 활용을 위해 Proxmox 환경에 대한 사전 이해가 필요하다.

각 방식은 사용자의 적용 환경과 목적, 자동화 범위, 관리 수준 등을 고려하여 선택될 수 있으며, 이를 통해 Terraform을 중심으로 한 IaC 활용은 On-premise 환경에서도 충분히 실현 가능함을 입증하였다.

## 참고문헌

[1] 이지은, 김수린, 한채림, 이호정, 김형종. (2024). IaC 기반 클라우드 인프라 자동화 도구 개발 및 보안 강화 방안 연구. 정보보호학회논문지, 34(6), 1273-1282.