MAAS와 Ansible을 활용한 On-Premise 인프라스트럭처 자동화 구현

박민혁, 오명훈
호남대학교 컴퓨터공학과
20210288@my.honam.ac.kr, mhoh@honam.ac.kr

Implementation of On-Premise Infrastructure Automation with MAAS and Ansible

Min-Hyuk Park, Myeong-Hoon Oh
Dept. of Computer Engineering, Honam University

요 약

본 논문은 베어메탈 서버부터 애플리케이션 레벨까지 일관된 자동화 인프라스트럭처을 구현하기 위해 MAAS(Metal as a Service)와 Ansible을 통합하는 방법론을 제안한다. 두 도구의 연계를 통해 서버의 초기 설치부터 서비스 배포까지 전체 과정을 코드로 관리할 수 있으며, 반복 가능하고 표준화된 인프라 환경을 구축하였다. MAAS의 물리적 서버 프로비저닝 기능과 Ansible의 구성 관리 능력을 결합하여 온프레미스 환경에서도 클라우드 수준의 자동화를 구현한다.

Ⅰ. 서론

현재 베어메탈 서버 환경에서도 클라우드와 같은 수준의 관리 효율성과와 인프라 자동화를 구현하려는 노력이 지속되고 있다. 이러한 환경에서 IaC(Infrastructure as Code) 방식이 주목받고 있으며, 코드를 통해 인프라스트럭처를 정의하고 프로비저닝하는 방식이 표준으로 자리잡고 있다[1].

기존의 인프라스트럭처 관리 접근법에는 여러 한계가 존재한다. 물리적 서버의 프로비저닝과 소프트웨어 구성 관리가 분리된 도구와 프로세스로 이루어지면서 통합적인 관리가 어렵고, 수작업으로 인한 오류와 비효율이 발생한다. 특히 대규모 인프라 환경에서는 일관된 구성과 신속한 변경 적용이 어려우며, 서버 자원의 효율적인 활용과 모니터링에도 한계가 있다. 또한 물리적 인프라와 가상 인프라를 통합적으로 관리할 수 있는 솔루션의 부재로 관리 복잡성이 증가하고 있다[2].

Ansible은 Python 모듈 및 SSH를 사용하여 원격 호스트에 명령을 수행할 수 있는 구성 관리 도구이다[3]. 별도의 에이전트 설치 없이 SSH 접속만으로 운영이 가능하며, 멱등성(Idempotency)을 보장하여 동일한 작업을 여러 번 실행해도 동일한 결과를 얻을 수 있다.

Ansible의 주요 구성 요소로는 Control Node, Managed Node, Inventory, Playbook, Module 등이 있다[1]. Control Node는 Ansible을 실행하는 노드이며, Managed Node는 Ansible로 관리되는 서버이다. Inventory는 관리 대상 호스트 정보가 포함된 파일이고, Playbook은 YAML 형식으로 작성된 자동화 작업 단위이다[4].

MAAS는 물리 서버 자원을 클라우드 인스턴스처럼 관리하고 자동 프로비저닝할 수 있도록 설계된 오픈소스 도구이다. 기존의 물리 서버 구축 과정이 수동적이고 복잡했던 것에 반해, MAAS는 이러한 과정을 자동화하여 물리 자원을 효율적으로 활용할 수 있게 한다.

MAAS가 물리서버의 프로비저닝과 네트워크 구성을 담당하고, Ansible이 소프트웨어 설치와 애플리케이션 배포 등 상위 레이어 자동화를 담당하여, 인프라 전체 라이프사이클을 자동화할 수 있다. 또한 코드 기반의 정의와 자동화로 환경 간 일관성을 보장하고, 반복 작업에 따른 오류를 최소화한다.

본 연구는 MAAS(Metal as a Service)[5]와 Ansible[6]을 통합하여 베어메탈 서버부터 애플리케이션 레벨까지 일관된 자동화 인프라스트럭처를 구현하는 방법론을 제안한다. MAAS의 강력한 물리적 서버 프로비저닝 기능과 Ansible의 유연한 구성 관리 능력을 결합함으로써, 클라우드 환경에서의 자동화 효율성을 온프레미스 환경에서도 구현하고자 한다.

Ⅱ. 본론

구현 환경은 1대의 네트워크 스위치, 1대의 관리 서버, 3대의 베어메탈 서버로 구성되었다. 관리 서버에는 MAAS와 Ansible이 설치되었으며, 베어메탈 서버는 동일한 성능을 가진 워크 스테이션으로 구성하였다.

그림 1은 본문의 구성 환경을 나타내고 있다.

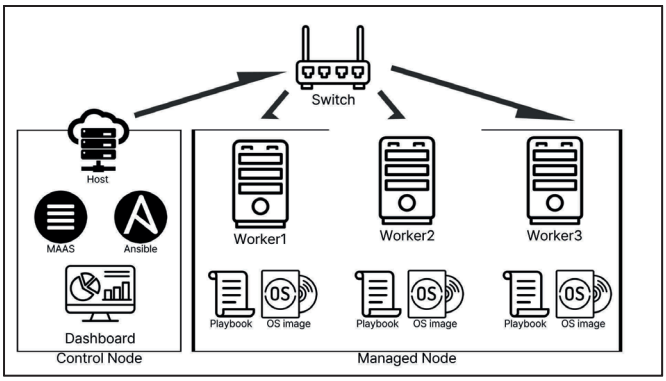

그림 1. 노드 구성 환경

Control Node에 MAAS를 설치하고 데이터베이스를 구성하여 MAAS Dashboard를 구현하였다. 워크스테이션 부팅옵션에서 PXE-Booting을 선택하면 같은 LAN을 공유하는 네트워크 환경에서 자동으로 Cloud-init가 Control Node의 MAAS를 식별하고 OS 이미지를 받아와 자동으로 OS를 구성하기 시작한다. 대시보드에선 구성을 마친 노드들의 세부설정을 해줄 수 있다. 세부설정까지 마친 노드들은 그림 2와 같이 Deployed 상태가 되어 원격으로 노드의 전원이나 네트워크 등을 컨트롤 할 수 있게 된다.

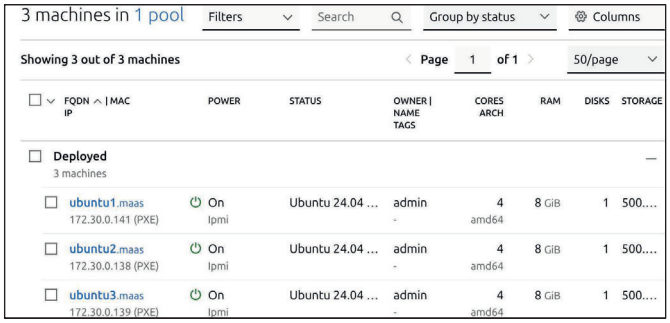

그림 2. MAAS Dashboard

Ansible-playbook은 자주 사용하는 명령어들을 자동화 스크립트로 구현하였다. Playbook은 YAML(YAML Ain't Markup Language) 형식으로 작성된다.

```
#APT Repository Reset Playbook(Temporary)

---
- name: Reset APT sources to default (Ubuntu 24.04 - Noble Numbat)
  hosts: nodes
  become: yes
  tasks:
    - name: Backup current sources.list
      copy:
        src: /etc/apt/sources.list
        dest: /etc/apt/sources.list.backup
        remote_src: yes
        force: no

    - name: Replace sources.list with default Ubuntu 24.04 (noble) sources
      copy:
        dest: /etc/apt/sources.list
        content: |
          deb http://archive.ubuntu.com/ubuntu noble main restricted universe multiverse
          deb http://archive.ubuntu.com/ubuntu noble-updates main restricted universe multiverse
          deb http://archive.ubuntu.com/ubuntu noble-backports main restricted universe multiverse
          deb http://security.ubuntu.com/ubuntu noble-security main restricted universe multiverse
```

그림 3. APT Reset Playbook

그림 3은 Ansible-playbook의 예제이다. 첫 번째 란을 보면 hosts, become, tasks가 있는데 hosts는 플레이북이 실행될 노드들을 칭하고, become은 root 권한을 주는 것을 말한다. tasks는 실제로 실행될 명령어들이다. 실제로 Playbook을 실행하게 되면, task가 위에서 아래로 차례대로 실행된다.

Playbook에 문법적인 오류가 있어 서버 장애 발생이 되더라도, 그림 2와 같은 대시보드에서 신속하게 신규 서버를 프로비저닝하고 Ansible로 기존 환경과 동일하게 설정할 수 있다. 서버 추가-삭제 및 확장 작업도 자동화되어 수작업 부담을 크게 줄일 수 있었다.

Control Node에서 Ansible-playbook을 작성 후, 3대의 Managed Node에 SSH를 통해 접근하여 Playbook을 바탕으로 각 노드마다 tasks에 있는 작업들을 한 줄씩 차례대로 진행하여 APT 저장소를 초기화하는 과정을 실시간으로 확인할 수 있다.

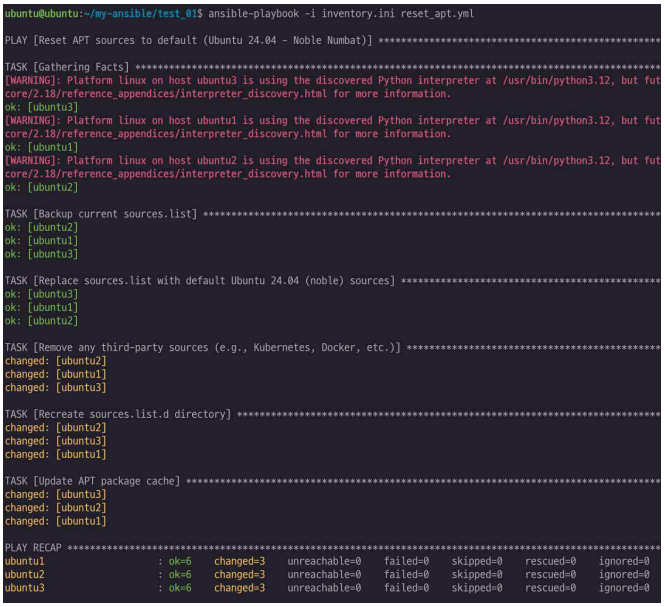

그림 4. Playbook Tasks

그림 4는 실제로 그림 3에서 작성된 Playbook을 실행해본 사진이다. Playbook이 한 줄씩 차례대로 실행되는 것과 경고 메시지 등을 확인할 수 있다. 또한 수작업으로 노드 하나하나 작업을 진행하는 것보다 작업 소요시간을 줄일 수 있었다. 단일 노드를 수작업으로 작업하는데 걸리는 시간이 10분이었고, Ansible 활용 당시 노드 3대 기준 똑같이 10분 정도 소요되었다. 단순히 수치화만 해도 66%의 시간을 단축한다. 이는 Ansible이 작업해야 할 노드가 많아질수록 효과적임을 보여준다.

Ⅲ. 결론

온프레미스 환경 구축 및 관리의 전통적인 어려움, 수동 작업으로 인한 시간 소모, 오류 발생 가능성, 그리고 확장성 부족 문제를 해결하고 클라우드 환경과 유사한 수준의 민첩성과 자동화를 달성하기 위한 방안으로 MAAS와 Ansible의 통합 활용을 제시하였다. (서버 추가 및 삭제, task 구성 확인에 관한 내용) 향후 연구로는 특정 클라우드 스택 구축 시 MAAS와 Ansible의 구체적인 통합 구현 사례를 심층적으로 분석하거나, 이러한 자동화 파이프라인을 CI/CD 워크플로우와 연동하여 인프라 변경 및 애플리케이션 배포까지 완전 자동화하는 방안에 대한 연구를 진행할 수 있을 것이다.

참 고 문 헌

[1] Ansible 기본 개념, https://chan-it-note.tistory.com/128, 2024
[2] "A Multi-Tenant System for 5/6G Testbed as-a-Service" https://www.themoonlight.io/ko/review/a-multi-tenant-system-for-56g-testbed-as-a-service, 2025
[3] "[Ansible] 기본 개념 정리" https://nyyang.tistory.com/188, 2024
[4] "Ansible Playbook: 자동화 태스크 설계 및 최적화 가이드" https://www.redhat.com/ko/topics/automation-and-management/ansible-playbookilan, 2023
[5] "Canonical. "MAAS | Metal as a Service". https://maas.io/
[6] "Red Hat. "Ansible Automation Platform." https://www.ansible.com/