2025-11-24 개발 세션 기록

작업: Issue #6 - Implement cloud-init for noVNC installation

작업 요약

목표

cloud-init 스크립트로 VM 부팅 시 Ubuntu Desktop + VNC + noVNC 자동 설치 및 설정.

완료된 작업

- **Branch:** feature/issue-6-cloud-init
- **Commits:** 1개 (진행 중)
- **PR:** 아직
- **Issue:** #6 진행 중

사전 논의 및 결정사항

1. VM 사이즈 변경

- **Before:** `Standard_B1s` (1 vCPU, 1GB RAM)

- **After:** `Standard_B2s` (2 vCPU, 4GB RAM)
- **이유:** Ubuntu Desktop + VNC 실행하기엔 1GB 부족

2. VNC 서버 선택

	TightVNC Server (Linux)	TigerVNC Server
마지막 업데이트	2009년 (v1.3.10)	2024년 (v1.15.0)
패키지명	`tightvncserver`	`tigervnc-standalone-server`
TLS 암호화	없음	내장

- **결정:** `tigervnc-standalone-server` 사용
- **이유:** TightVNC Linux 버전은 16년째 업데이트 없음

3. VNC 비밀번호

- **현재:** 고정값 `unity123`
- **나중에 (Phase 6):** GitHub Secret에서 `VNC_PASSWORD` 환경변수로 사용자 지정

4. USERNAME 변수화

- **이유:** 클라우드마다 기본 사용자명이 다름
  - Azure: `azureuser`
  - AWS: `ubuntu`
  - GCP: 사용자 지정
- **결정:** `{{USERNAME}}` placeholder로 만들어서 나중에 멀티 클라우드 지원 가능하게

아키텍처 이해

noVNC 전체 흐름

```
[사용자 브라우저] → [noVNC/websockify] → [VNC 서버] → [데스크톱 환경]
     (port 6080)        (port 6080)       (port 5901)      (xfce4)

```

각 패키지 역할

패키지	역할
`xfce4`	경량 데스크톱 환경 (GUI)
`tigervnc-standalone-server`	VNC 서버, 데스크톱 화면을 네트워크로 전송

`novnc` + `websockify`	VNC를 웹소켓으로 변환, 브라우저 접속 가능하게 함

VNC 포트 규칙

- VNC 표준 포트: 5900
- 디스플레이 :1 → 5901 (5900 + 1)
- 디스플레이 :2 → 5902 (5900 + 2)

Commit 1: Add cloud-init module and NSG

생성된 파일

lib/config.js

```
export const config = {
    vm: {
        adminUsername: 'azureuser',
        adminPassword: 'Azure123456!',
        size: 'Standard_B2s'
    },
    vnc: {
        password: 'unity123',
        port: 5901,
        geometry: '1280x800'
    },
    novnc: {
        port: 6080
    }
};

```

lib/cloud-init-template.yaml

```
#cloud-config
# cloud-init에게 "이건 설정 파일이야" 알려주는 필수 헤더
# 이게 없으면 cloud-init이 인식 못 함

package_update: true
# apt update 실행
# 패키지 설치 전에 패키지 목록 최신화

packages:
  - xfce4
  # 경량 데스크톱 환경
  # GNOME보다 가벼워서 VM에 적합
  # 이게 있어야 GUI "화면"이 생김

  - xfce4-goodies
  # xfce4 추가 유틸리티 (파일 관리자, 터미널 등)

  - tigervnc-standalone-server
  # VNC 서버
  # 데스크톱 화면을 네트워크로 전송
  # tightvncserver 대신 사용 (2024년 기준 활발히 유지보수 중)

  - novnc
  # 웹 기반 VNC 클라이언트
  # HTML/JS로 구성된 웹 페이지 제공

  - websockify
  # VNC(TCP) → WebSocket 변환 프록시
  # 브라우저는 WebSocket만 지원하므로 필요

```

```yaml
write_files:
  - path: /home/{{USERNAME}}/.vnc/xstartup
    # VNC 서버 시작 시 실행되는 스크립트 경로
    # VNC 세션에서 "뭘 보여줄지" 정의

    permissions: '0755'
    # 파일 권한 설정
    # 0755 = rwxr-xr-x
    # 소유자: 읽기+쓰기+실행 (7)
    # 그룹: 읽기+실행 (5)
    # 기타: 읽기+실행 (5)
    # 실행 권한 없으면 VNC 서버가 이 스크립트 실행 못 함

    content: |
      #!/bin/bash
      # bash 스크립트로 실행

      unset SESSION_MANAGER
      unset DBUS_SESSION_BUS_ADDRESS
      # 기존 데스크톱 세션 환경변수 제거
      # 안 하면 VNC 세션이 기존 세션이랑 충돌 가능

      xrdb $HOME/.Xresources
      # X 서버 리소스 설정 로드 (폰트, 색상 등)
      # 없어도 되지만 관례적으로 포함

      startxfce4 &
      # xfce4 데스크톱 환경 시작
      # & = 백그라운드 실행
      # 이게 없으면 VNC 접속해도 회색 화면만 보임

runcmd:
  # 모든 패키지 설치 완료 후 순서대로 실행되는 명령어들
  # cloud-init은 root 권한으로 실행됨

  - mkdir -p /home/{{USERNAME}}/.vnc
  # VNC 설정 폴더 생성
  # -p: 상위 폴더 없으면 자동 생성

  - chown -R {{USERNAME}}:{{USERNAME}} /home/{{USERNAME}}/.vnc
  # 폴더 소유자를 azureuser로 변경
  # cloud-init은 root로 실행되므로 이거 안 하면 azureuser가 접근 못 함
  # -R: 하위 폴더/파일 전부 적용

  - echo '{{VNC_PASSWORD}}' | vncpasswd -f >
/home/{{USERNAME}}/.vnc/passwd
  # VNC 비밀번호 설정
  # echo로 비밀번호 출력 → vncpasswd -f로 암호화 → passwd 파일에 저장
  # -f: stdin에서 비밀번호 읽기 (interactive 모드 대신)

  - chmod 600 /home/{{USERNAME}}/.vnc/passwd
  # 비밀번호 파일 권한 설정
  # 600 = rw-------
  # 소유자만 읽기+쓰기, 다른 사람은 접근 불가
  # 보안상 필수

  - chown {{USERNAME}}:{{USERNAME}} /home/{{USERNAME}}/.vnc/passwd
  # 비밀번호 파일 소유자 변경

  - su - {{USERNAME}} -c 'vncserver :1 -geometry 1280x800 -depth 24'
  # VNC 서버 시작
  # su - {{USERNAME}} -c: azureuser 권한으로 명령어 실행
  # vncserver :1: 디스플레이 번호 1번으로 시작 (포트 = 5900 + 1 = 5901)
  # vncserver는 default로 포트 번호 5900으로 시작함.
  # -geometry 1280x800: 가상 데스크톱 해상도
  # -depth 24: 색상 깊이 24비트 (트루컬러, 1600만 색상)

  - websockify --web=/usr/share/novnc 6080 localhost:5901 &
  # noVNC 웹소켓 프록시 시작
  # --web=/usr/share/novnc: noVNC 웹 파일 경로 (HTML/JS/CSS)
  # 6080: 웹소켓 서버 포트 (브라우저가 접속하는 포트)
```

```
  # localhost:5901: 연결할 VNC 서버 주소 (위에서 시작한 VNC)
  # &: 백그라운드 실행
```

최종 통신 흐름

- 사용자 브라우저
- ↓ HTTP/WebSocket (port 6080)
- websockify (noVNC 웹 제공 + WebSocket→TCP 변환)
- ↓ TCP (port 5901)
- tigervnc-standalone-server (VNC 서버)
- ↓ X11 프로토콜
- xfce4 (데스크톱 환경)

lib/cloudInitScript.js

```
import { readFileSync } from 'fs';
import { fileURLToPath } from 'url';
import { dirname, join } from 'path';
import { config } from './config.js';

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);

export function generateCloudInitScript(
    vncPassword = config.vnc.password,
    username = config.vm.adminUsername
) {
    const templatePath = join(__dirname, 'cloud-init-template.yaml');
    let template = readFileSync(templatePath, 'utf8');

    template = template.replace(/{{VNC_PASSWORD}}/g, vncPassword);
    template = template.replace(/{{USERNAME}}/g, username);

    return Buffer.from(template).toString('base64');
}
```

vmManager.js 수정사항

1. **import 추가**

```
import { config } from './config.js';

```

2. **createNetworkSecurityGroup() 함수 추가**

```
async function createNetworkSecurityGroup(networkClient, resourceGroupName, location, vmName) {
    const nsgName = `${vmName}-nsg`;

    console.log('Creating network security group...');
    const nsgParams = {
        location: location,
        securityRules: [
            {
                name: 'allow-novnc',
                protocol: 'Tcp',
                sourcePortRange: '*',
                destinationPortRange: String(config.novnc.port),
                sourceAddressPrefix: '*',
                destinationAddressPrefix: '*',
                access: 'Allow',
                priority: 100,
                direction: 'Inbound'
```

```
            }
        ]
    };

    const nsg = await networkClient.networkSecurityGroups.beginCreateOrUpdateAndWait(
        resourceGroupName,
        nsgName,
        nsgParams
    );

    console.log('Network security group created');
    return nsg;
}

```

3. **createNetworkInterface() 수정**

- `nsg` 파라미터 추가
- NIC에 NSG 연결

4. **provisionVM() 수정**

```
const vnet = await createVirtualNetwork(...);
const publicIP = await createPublicIP(...);
const nsg = await createNetworkSecurityGroup(...);
const nic = await createNetworkInterface(..., vnet, publicIP, nsg);

```

5. **VM 사이즈 변경**

- `Standard_B1s` → `config.vm.size` (Standard_B2s)

6. **customData (임시 주석)**

- NSG 먼저 테스트하기 위해 주석 처리

테스트 결과

```
[2025-11-24T16:39:58.747Z] Creating network security group...
[2025-11-24T16:39:59.394Z] Network security group created
[2025-11-24T16:39:59.396Z] Creating network interface...
[2025-11-24T16:40:02.368Z] Network interface created
[2025-11-24T16:40:02.369Z] Network resources ready
```

Azure Portal 확인:

- NSG 생성됨: `vm-xxx-nsg`
- Inbound rule: `allow-novnc` (port 6080, TCP, Allow)
- NIC에 NSG 연결됨

주요 학습 포인트

1. cloud-init 구조

```
#cloud-config           # 필수 헤더
package_update: true    # apt update
packages:               # apt install
write_files:            # 파일 생성
runcmd:                 # 설치 후 명령어 실행
```

2. Base64 인코딩 이유

- Azure `customData`가 Base64 문자열만 받음

- JSON 전송 시 특수문자/줄바꿈 문제 방지

<details>
<summary>UTF-8과 Base64 인코딩</summary>

UTF-8이란: 사람이 읽을 수 있는 일반 텍스트 인코딩.

Base64란: 바이너리 데이터를 ASCII 문자(A-Z, a-z, 0-9, +, /)로만 표현하는 인코딩.

변환 예시:

```
UTF-8 (원본):
#cloud-config
packages:
  - xfce4

Base64 (변환 후):
I2Nsb3VkLWNvbmZpZwpwYWNrYWdlczoKICAtIHhmY2U0Cg==
```

왜 Base64가 필요한가:

Azure API는 JSON으로 데이터를 보내는데, JSON 안에 이런 문자가 있으면 문제가 생김:

- 줄바꿈 (`\n`)
- 따옴표 (`"` , `'`)
- 특수문자 (`#` , `:` , `-`)

```
// 문제가 되는 경우 (UTF-8 그대로)
{
  "customData": "#cloud-config
packages:
  - xfce4"   // ← JSON 파싱 에러!
}

// 안전한 경우 (Base64)
{
  "customData": "I2Nsb3VkLWNvbmZpZwpwYWNrYWdlczoKICAtIHhmY2U0Cg=="
}
```

Azure에서 처리 과정:

```
1. 우리 코드: YAML → Base64 인코딩 → Azure API 전송
2. Azure: Base64 문자열 받아서 VM에 저장
3. VM 부팅: Base64 → 디코딩 → 원래 YAML 복원 → cloud-init 실행
```

코드에서:

```
// Buffer.from(template) - UTF-8 문자열을 바이너리로
// .toString('base64') - 바이너리를 Base64 문자열로
return Buffer.from(template).toString('base64');
```

</details>

3. __dirname 사용 이유

- Node.js에서 파일 읽을 때 실행 위치 기준이 됨
- `__dirname` 으로 항상 파일 위치 기준 경로 사용

4. 테스트 분리의 중요성

- cloud-init + NSG 동시에 하다가 에러 발생
- NSG 먼저 테스트 → 확인 후 커밋 → cloud-init 추가
- 한 번에 너무 많이 바꾸지 말 것

현재 파일 구조

```
lib/
├── config.js                 (신규)
├── cloud-init-template.yaml  (신규)
├── cloudInitScript.js        (신규)
└── vmManager.js              (수정)
```

다음 작업: Commit 2

할 일

1. `customData` 주석 해제
2. `generateCloudInitScript` import 및 호출
3. VM 생성 테스트
4. noVNC 접속 테스트 (`http://{PUBLIC_IP}:6080`)

테스트 체크리스트

- [ ] VM 생성 완료
- [ ] cloud-init 실행 완료 (5-10분 소요)
- [ ] `http://{PUBLIC_IP}:6080` 접속
- [ ] VNC 비밀번호 입력 (`unity123`)
- [ ] xfce4 데스크톱 화면 표시

Phase 3 진행 상황

완료:

- ✅ Issue #5: Public IP allocation
- ✅ Issue #11: Refactor VM provisioning

진행 중:

- 🔄 Issue #6: cloud-init for noVNC (Commit 1 완료, Commit 2 진행 예정)

남은 작업:

- ⏳ Issue #9: Test noVNC web access

작업 시간

- 사전 논의 및 설계: ~30분
- 코드 작성: ~40분
- 테스트 및 디버깅: ~20분
- **총 작업 시간:** ~1.5시간

Status: Commit 1 완료, Commit 2 진행 예정 🔄