# Red Hat Enterprise Linux 8

## 사용자 지정 RHEL 시스템 이미지 구성

Red Hat Enterprise Linux 8에서 이미지 빌더를 사용하여 사용자 지정 시스템 이미지 생성

Last Updated: 2022-08-29

# Red Hat Enterprise Linux 8 사용자 지정 RHEL 시스템 이미지 구성

Red Hat Enterprise Linux 8에서 이미지 빌더를 사용하여 사용자 지정 시스템 이미지 생성

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

## 법적 공지



## 초록

이미지 빌더는 설치 디스크, 가상 머신, 클라우드 벤더별 이미지 등 배포 준비 시스템 이미지를 생성하는 툴입니다. 이미지 빌더를 사용하면 각 출력 유형에 필요한 특정 구성을 제거하므로 수동 프로시저와 비교하여 이러한 이미지를 더 빠르게 생성할 수 있습니다. 이 문서에서는 Image Builder를 설정하고 이미지를 생성하는 방법을 설명합니다.

# 차례

# 보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

# RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. 문서를 개선할 수 있는 방법에 관해 알려주십시오.

- 특정 문구에 대한 간단한 의견 작성 방법은 다음과 같습니다.
  1. 문서가 *Multi-page HTML* 형식으로 표시되는지 확인합니다. 또한 문서 오른쪽 상단에 **피드백** 버튼이 있는지 확인합니다.
  2. 마우스 커서를 사용하여 주석 처리하려는 텍스트 부분을 강조 표시합니다.
  3. 강조 표시된 텍스트 아래에 표시되는 **피드백 추가** 팝업을 클릭합니다.
  4. 표시된 지침을 따릅니다.
- Bugzilla를 통해 피드백을 제출하려면 새 티켓을 생성합니다.
  1. Bugzilla 웹 사이트로 이동하십시오.
  2. 구성 요소로 **문서**를 사용합니다.
  3. **설명** 필드에 문서 개선을 위한 제안 사항을 기입하십시오. 관련된 문서의 해당 부분 링크를 알려주십시오.
  4. **버그 제출**을 클릭합니다.

# 1장. 이미지 빌더 설명

## 1.1. 이미지 빌더 소개

이미지 빌더를 사용하여 클라우드 플랫폼에 배포할 수 있도록 준비된 시스템 이미지를 포함하여 Red Hat Enterprise Linux의 사용자 지정 시스템 이미지를 생성할 수 있습니다. 이미지 빌더는 각 출력 유형에 대한 설정 세부 정보를 자동으로 처리하고 따라서 이미지 생성의 수동 방법보다 사용하기 쉽고 빠르게 작업할 수 있습니다. **composer-cli** 툴의 명령줄 인터페이스 또는 RHEL 웹 콘솔에서 그래픽 사용자 인터페이스를 통해 Image Builder 기능에 액세스할 수 있습니다.

Red Hat Enterprise Linux 8.3부터 **osbuild-composer** 백엔드는 **lorax-composer** 를 대체합니다. 새 서비스는 이미지 빌드를 위한 REST API를 제공합니다. 결과적으로 사용자는 보다 안정적인 백엔드 및 보다 예측 가능한 출력 이미지를 활용할 수 있습니다.

이미지 빌더는 시스템 서비스 **osbuild-composer** 로 실행됩니다. 다음 두 인터페이스를 통해 이 서비스와 상호 작용할 수 있습니다.

- CLI 툴 **composer-cli** - 터미널에서 명령을 실행합니다. 이 방법이 선호됩니다.
- RHEL 웹 콘솔용 GUI 플러그인.

## 1.2. 이미지 빌더 용어

**Quarkus**

Makefile은 시스템의 일부가 될 패키지 및 사용자 지정을 나열하여 사용자 지정 시스템 이미지를 정의합니다. Makefile은 편집할 수 있으며 버전이 지정됩니다. 시스템 이미지가 capabilities에서 생성되는 경우 이미지는 RHEL 웹 콘솔의 이미지 빌더 인터페이스의 panic과 연관됩니다.
지멘션은 TOML 형식의 일반 텍스트로 사용자에게 표시됩니다.

**compose**

composes는 시스템 이미지의 개별 빌드이며, 특정 Makefile의 특정 버전을 기반으로 합니다. 용어로 작성은 시스템 이미지, 생성, 입력, 메타데이터 및 프로세스 자체의 로그를 나타냅니다.

**사용자 정의**

사용자 지정은 패키지가 아닌 시스템의 사양입니다. 여기에는 사용자, 그룹 및 SSH 키가 포함됩니다.

## 1.3. 이미지 빌더 출력 형식

이미지 빌더는 다음 표에 표시된 여러 출력 형식으로 이미지를 생성할 수 있습니다. 지원되는 유형을 확인하려면 명령을 실행합니다.

```
# composer-cli compose types
```

**표 1.1. 이미지 빌더 출력 형식**

| 설명 | CLI 이름 | 파일 확장자 |
|---|---|---|
| QEMU QCOW2 이미지 | **qcow2** | **.qcow2** |
| TAR 아카이브 | **tar** | **.tar** |

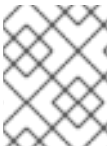
| 설명 | CLI 이름 | 파일 확장자 |
|---|---|---|
| Amazon 머신 이미지 디스크 | **ami** | **.raw** |
| Azure Disk Image | **vhd** | **.vhd** |
| VMware Virtual Machine Disk | **vmdk** | **.vmdk** |
| OpenStack | **openstack** | **.qcow2** |
| 에지 커밋용 RHEL | **edge-commit** | **.tar** |
| 에지 컨테이너용 RHEL | **edge-container** | **.tar** |
| Edge 설치 프로그램용 RHEL | **edge-installer** | **.iso** |
| RHEL for Edge Raw | **edge-raw-image** | **.tar** |
| RHEL for Edge Simplified Installer | **edge-simplified-installer** | **.iso** |
| ISO 이미지 | **image-installer** | **.iso** |

## 1.4. 이미지 빌더 시스템 요구 사항

이미지 빌더가 실행되는 환경(예: 전용 가상 머신)은 다음 표에 나열된 요구 사항을 충족해야 합니다.

**표 1.2. 이미지 빌더 시스템 요구 사항**

| 매개변수 | 최소 필수 값 |
|---|---|
| 시스템 유형 | 전용 가상 머신 |
| 프로세서 | 2개의 코어 |
| 메모리 | 4GiB |
| 디스크 공간 | **/var** 파일 시스템에서 20GiB의 여유 공간 |
| 액세스 권한 | 관리자 수준(root) |
| 네트워크 | 인터넷 연결 |



**참고**

인터넷 연결이 전제 조건이 아닙니다. Red Hat CDN에 연결하지 않도록 재구성하는 경우 격리된 네트워크에서 이미지 빌더를 사용할 수 있습니다.

# 2장. 이미지 빌더 설치

이미지 빌더를 사용하기 전에 가상 머신에 이미지 빌더를 설치해야 합니다.

## 2.1. 가상 머신에 이미지 빌더 설치

전용 가상 머신에 이미지 빌더를 설치하려면 다음 단계를 따르십시오.

**사전 요구 사항**

- 가상 머신에 연결합니다.
- 이미지 빌더용 가상 시스템을 설치하고, Red Hat Subscription Manager(RHSM) 또는 Red Hat Satellite에 등록하고 실행 중이어야 합니다.

**절차**

1. 가상 머신에 이미지 빌더 및 기타 필요한 패키지를 설치합니다.
   - **osbuild-composer** - RHEL 8.3에서 지원됨
   - **composer-cli**
   - **cockpit-composer**
   - **bash-completion**

   ```
   # yum install osbuild-composer composer-cli cockpit-composer bash-completion
   ```

   웹 콘솔은 *cockpit-composer* 패키지의 종속성으로 설치됩니다.

2. 재부팅할 때마다 이미지 빌더를 시작하도록 활성화합니다.

   ```
   # systemctl enable --now osbuild-composer.socket
   # systemctl enable --now cockpit.socket
   ```

   **osbuild-composer** 및 **cockpit** 서비스는 첫 번째 액세스 시 자동으로 시작됩니다.

3. **composer-cli** 명령의 자동 완성 기능이 재부팅 없이 즉시 작동을 시작하도록 쉘 구성 스크립트를 로드합니다.

   ```
   $ source /etc/bash_completion.d/composer-cli
   ```

**중요**

**osbuild-composer** 패키지는 Red Hat Enterprise Linux 8.3 이후부터 모든 새로운 기능의 기본 설정 및 중점을 둔 새로운 백엔드 엔진입니다. 이전 백엔드 **lorax-composer** 패키지는 더 이상 사용되지 않는 것으로 간주되며 나머지 Red Hat Enterprise Linux 8 라이프 사이클에 대한 일부 수정 사항만 제공되며 향후 주요 릴리스에서 생략될 예정입니다. osbuild-composer 대신 **lorax-composer** 를 제거하는 것이 좋습니다.

**검증**

시스템 저널을 사용하여 이미지 빌더 서비스 활동을 추적할 수 있습니다. 또한 파일에서 로그 메시지를 찾을 수 있습니다.

- 역추적에 대한 저널 출력을 찾으려면 다음 명령을 실행합니다.

```
$ journalctl | grep osbuild
```

- 원격 또는 로컬 작업자를 모두 표시하려면 다음을 수행합니다.

```
$ journalctl -u osbuild-worker*
```

- 실행 중인 서비스를 표시하려면 다음을 수행합니다.

```
$ journalctl -u osbuild-composer.service
```

## 2.2. LORAX-COMPOSER IMAGE BUILDER 백엔드로 되돌리기

**osbuild-composer** 백엔드는 더 많은 확장이 가능하지만 현재 이전의 **lorax-composer** 백엔드와 기능 패리티를 제공하지 않습니다.

이전 백엔드로 되돌리려면 다음 단계를 따르십시오.

**사전 요구 사항**

- **osbuild-composer** 패키지를 설치했습니다.

**절차**

1. osbuild-composer 백엔드를 제거합니다.

```
# yum remove osbuild-composer
```

2. **/etc/yum.conf 파일에서 osbuild-composer** 패키지에 대한 exclude 항목을 추가합니다.

```
# cat /etc/yum.conf
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
best=True
skip_if_unavailable=False
exclude=osbuild-composer
```

3. **lorax-composer** 패키지를 설치합니다.

```
# yum install lorax-composer
```

4. 재부팅할 때마다 **lorax-composer** 서비스를 활성화 및 시작하여 시작합니다.

```
# systemctl enable --now lorax-composer.socket
# systemctl start lorax-composer
```

추가 리소스

- Red Hat 지원 케이스 작성 .

# 3장. 리포지토리 관리

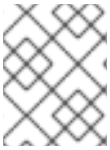
## 3.1. IMAGE BUILDER 기본 시스템 리포지토리

**osbuild-composer** 백엔드는 **/etc/yum.repos.d/** 디렉터리에 있는 시스템 리포지토리를 상속하지 않습니다. 대신 **/usr/share/osbuild-composer/repositories** 디렉터리에 정의된 자체 공식 리포지토리 집합이 있습니다. 공식 리포지토리를 재정의하려면 **/etc/osbuild-composer/repositories** 에 재정의를 정의해야 합니다. 이 디렉토리는 사용자 정의 덮어쓰기용이며 여기에 있는 파일은 **/usr** 디렉토리에 있는 파일보다 우선합니다.

구성 파일은 **/etc/yum.repos.d/** 의 파일에서 알려진 일반 YUM 리포지토리 형식이 아닙니다. 대신 간단한 JSON 파일입니다.

## 3.2. 시스템 리포지토리 덮어쓰기

다음 단계를 사용하여 **/etc/osbuild-composer/repositories** 디렉토리에서 리포지토리 재정의를 구성할 수 있습니다.



**참고**

RHEL 8.5 릴리스 이전에는 리포지토리 덮어쓰기 이름이 **rhel-8.json** 입니다. RHEL 8.5부터 이름은 마이너 버전 **rhel-84.json**,**rhel-85.json** 등이 적용됩니다.

**사전 요구 사항**

- 호스트 시스템에서 액세스할 수 있는 사용자 지정 리포지토리가 있습니다.

**절차**

1. 사용하려는 저장소 덮어쓰기가 포함된 디렉터리를 생성합니다.

```
$ sudo mkdir -p /etc/osbuild-composer/repositories
```

2. 다음 구조를 사용하여 JSON 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
{
  "<ARCH>": [
    {
      "name": "baseos",
      "metalink": "",
      "baseurl": "http://mirror.example.com/composes/released/RHEL-
8/8.0/BaseOS/x86_64/os/",
      "mirrorlist": "",
      "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (…)",
      "check_gpg": true,
      "metadata_expire": ""
    }
  ]
}
```

**metalink**,**mirrorlist** 또는 **baseurl** 중 하나만 지정합니다. 나머지 필드는 선택 사항입니다.

3. RHEL 버전에 해당하는 이름을 사용하여 JSON 파일을 저장합니다. 예를 들면 다음과 같습니다.

```
/etc/osbuild-composer/repositories/rhel-84.json
/etc/osbuild-composer/repositories/rhel-85.json
```

a. 또는 **/usr/share/osbuild-composer/** 에서 배포에 대한 JSON 파일을 복사하고 해당 콘텐츠를 수정할 수 있습니다.

i. 리포지토리 파일을 생성한 디렉터리에 복사합니다.

```
$ cp /usr/share/osbuild-composer/repositories/rhel-version.json /etc/osbuild-composer/repositories/
```

rhel-version.json을 RHEL 버전으로 교체합니다(예: rhel-8.json).

4. 텍스트 편집기를 사용하여 **rhel-8.json** 파일의 **baseurl** 경로를 편집합니다. 예를 들면 다음과 같습니다.

```
$ vi /etc/osbuild-composer/repositories/rhel-8.json
```

5. **osbuild-composer.service** 를 다시 시작하십시오.

```
$ sudo systemctl restart osbuild-composer.service
```

결과적으로 리포지토리는 **/etc/yum.repos.d/redhat.repo** 파일에서 복사되는 올바른 URL을 가리킵니다.

**추가 리소스**

- **osbuild-composer** 에 표시되지 않는 리포지토리에서 사용 가능한 최신 RPM입니다.

## 3.3. 서브스크립션을 지원하는 시스템 리포지토리 덮어쓰기

**osbuild-composer** 서비스는 **/etc/yum.repos.d/redhat.repo** 파일에 정의된 시스템 서브스크립션을 사용할 수 있습니다. **osbuild-composer** 에서 시스템 서브스크립션을 사용하려면 다음과 같은 리포지토리 재정의를 정의해야 합니다.

- **/etc/yum.repos.d/redhat.repo** 에 정의된 리포지토리와 동일한 **baseurl**.
- JSON 오브젝트에 정의된 **"rhsm": true** 의 값입니다.

**사전 요구 사항**

- 서브스크립션이 **/etc/yum.repos.d/redhat.repo**에 정의된 시스템
- 리포지터리 덮어쓰기가 생성되어 있습니다. 시스템 리포지토리 덮어쓰기를 참조하십시오.

**절차**

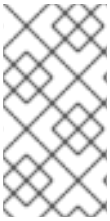
1. **/etc/yum.repos.d/redhat.repo 파일에서 baseurl 을 가져옵니다.**

```
[AppStream]
name = AppStream mirror example
baseurl = https://mirror.example.com/RHEL-8/8.0/AppStream/x86_64/os/
enabled = 1
gpgcheck = 0
sslverify = 1
```

```
sslcacert = /etc/pki/ca1/ca.crt
sslclientkey = /etc/pki/ca1/client.key
sslclientcert = /etc/pki/ca1/client.crt
metadata_expire = 86400
enabled_metadata = 0
```

2. 동일한 **baseurl** 및 set **rhsm** 을 true로 사용하도록 리포지토리 재정의를 구성합니다.

```
{
   "x86_64": [
      {
         "name": "AppStream mirror example",
         "baseurl": "https://mirror.example.com/RHEL-8/8.0/AppStream/x86_64/os/",
         "gpgkey": "-----BEGIN PGP PUBLIC KEY BLOCK-----\n\n (…)",
         "check_gpg": true,
         "rhsm": true
      }
   ]
}
```



**참고**

**osbuild-composer** 는 **/etc/yum.repos.d/** 에 정의된 리포지토리를 자동으로 사용하지 않습니다. **composer-cli** 를 사용하여 수동으로 시스템 리포지토리 재정의 또는 추가 **소스로** 지정해야 합니다. 시스템 리포지토리 재정의는 일반적으로 "BaseOS" 및 "AppStream" 리포지토리에 사용되는 반면 **composer-cli** 소스는 다른 모든 리포지토리에 사용됩니다.

**추가 리소스**

- 이미지 빌더는 호스트가 Satellite 6에 등록되어 있을 때 CDN 리포지토리를 사용합니다.

# 4장. 이미지 빌더 명령줄 인터페이스를 사용하여 시스템 이미지 생성

이미지 빌더는 사용자 지정 시스템 이미지를 생성하기 위한 툴입니다. 이미지 빌더를 제어하고 사용자 지정 시스템 이미지를 만들려면 현재 Image Builder를 사용하는 기본 방법인 명령줄 인터페이스를 사용합니다.

## 4.1. 이미지 빌더 명령줄 인터페이스

이미지 빌더 명령줄 인터페이스는 현재 Image Builder를 사용하는 기본 방법입니다. 웹 콘솔 인터페이스보다 더 많은 기능을 제공합니다. 이 인터페이스를 사용하려면 적절한 옵션 및 하위 명령을 사용하여 **composer-cli** 명령을 실행합니다.

명령줄 인터페이스의 워크플로는 다음과 같이 요약할 수 있습니다.

1. 요약 정보 내보내기 (*Save*) the plain text file
2. 텍스트 편집기에서 이 파일을 편집합니다.
3. Quarkus 텍스트 파일을 다시 이미지 빌더로 가져오기(*push*)
4. 작성을 실행하여 지침에서 이미지 빌드
5. 다운로드할 이미지 파일을 내보냅니다.

이 절차를 수행하기 위한 기본 하위 명령 외에도 **composer-cli** 명령은 구성된 Tangs 및 composes의 상태를 검사하는 많은 하위 명령을 제공합니다.

**composer-cli** 명령을 루트가 아닌 사용자로 실행하려면 사용자가 **weldr** 또는 **root** 그룹에 있어야 합니다.

- 사용자를 **weldr** 또는 **root** 그룹에 추가하려면 다음 명령을 실행합니다.

```
$ sudo usermod -a -G weldr user
$ newgrp weldr
```

## 4.2. 명령줄 인터페이스를 사용하여 이미지 빌더 롤업 생성

다음 절차에서는 명령줄 인터페이스를 사용하여 새 Image Builder 요건을 생성하는 방법을 설명합니다.

**절차**

1. 다음 콘텐츠를 사용하여 일반 텍스트 파일을 생성합니다.

```
name = "BLUEPRINT-NAME"
description = "LONG FORM DESCRIPTION TEXT"
version = "0.0.1"
modules = []
groups = []
```

*BLUEPRINT-NAME* 및 *LONG FORM DESCRIPTION TEXT* 를 devfile에 대한 이름 및 설명으로 바꿉니다.

*0.0.1*을 Semantic Versioning scheme에 따라 버전 번호로 바꿉니다.

2. Makefile에 포함하려는 모든 패키지에 대해 파일에 다음 행을 추가합니다.

-

```
[[packages]]
name = "package-name"
version = "package-version"
```

*package-name* 을 **httpd**, **gdb-doc** 또는 **coreutils** 와 같은 패키지 이름으로 바꿉니다.

사용할 *package-version* 을 버전으로 바꿉니다. 이 필드는 **dnf** 버전 사양을 지원합니다.

- 특정 버전의 경우 **8.6.0** 과 같은 정확한 버전 번호를 사용하십시오.
- 사용 가능한 최신 버전의 경우 별표 * 를 사용하십시오.
- 최신 마이너 버전의 경우 **8.***과 같은 형식을 사용하십시오.

3. 여러 가지 방법으로 탄력성을 사용자 지정할 수 있습니다. 이 예에서는 아래 단계를 수행하여 SMT(Simultaneous Multi Threading)를 비활성화할 수 있습니다. 사용 가능한 추가 사용자 지정은 지원되는 이미지 사용자 지정을 참조하십시오.

```
[customizations.kernel]
append = "nosmt=force"
```

4. 파일을 *BLUEPRINT-NAME*.toml로 저장하고 텍스트 편집기를 종료합니다.
5. credential을 푸시(가져오기)합니다.

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

*BLUEPRINT-NAME* 을 이전 단계에서 사용한 값으로 바꿉니다.

6. 기존journald를 나열하여509가 푸시되고 있는지 확인합니다.

```
# composer-cli blueprints list
```

7. 방금 추가한 NetNamespace 구성을 표시하려면 명령을 실행합니다.

```
# composer-cli blueprints show
```

8. Makefile 및 해당 종속 항목에 나열된 구성 요소 및 버전이 유효한지 확인합니다.

```
# composer-cli blueprints depsolve BLUEPRINT-NAME
```

**참고**

**composer-cli** 명령을 루트가 아닌 명령을 사용하여 이미지를 생성하려면 사용자를 **weldr** 또는 **root** 그룹에 추가합니다.

### 검증

이미지 빌더에서 사용자 지정 리포지토리에서 패키지를 제거할 수 없는 경우 다음 단계를 수행합니다.

- osbuild-composer 캐시를 제거합니다.

```
$ sudo rm -rf /var/cache/osbuild-composer/*
$ sudo systemctl restart osbuild-composer
```

-

**추가 리소스**

- **osbuild-composer** 는 사용자 정의 저장소에서 패키지를 제거할 수 없음
- 프록시 서버를 사용하여 사용자 지정된 RHEL 시스템 이미지 구성

## 4.3. 명령줄 인터페이스를 사용하여 이미지 빌더 승격 편집

명령줄 인터페이스에서 기존 이미지 빌더 청사진을 편집하려면 단계를 따르십시오.

**절차**

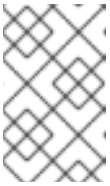
1. 배치를 로컬 텍스트 파일에 저장(export)합니다.

```
# composer-cli blueprints save BLUEPRINT-NAME
```

2. 텍스트 편집기로 *BLUEPRINT-NAME*.toml 파일을 편집하고 변경합니다.
3. 편집을 완료하기 전에 파일이 유효한 rootfs인지 확인하십시오.
   a. 존재하는 경우 이 행을 제거합니다.

```
packages = []
```

   b. 버전 번호를 늘립니다. Image Builder Provisioning 버전은 Semantic Versioning scheme을 사용해야 합니다. 또한 버전을 변경하지 않으면 **패치** 버전 구성 요소가 자동으로 증가합니다.
   c. 콘텐츠가 유효한 TOML 사양인지 확인합니다. 자세한 내용은 TOML 설명서 를 참조하십시오.



**참고**

TOML 문서는 커뮤니티 제품이며 Red Hat에서 지원하지 않습니다. 이 툴의 모든 문제는 https://github.com/toml-lang/toml/issues에서 보고할 수 있습니다.

4. 파일을 저장하고 텍스트 편집기를 종료합니다.
5. Credential을 Image Builder로 다시 푸시(Import)합니다.

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

**참고**

structures을 다시 Image Builder로 가져오려면 **.toml** 확장을 포함한 파일 이름을 다른 명령에서는journald 이름만 사용합니다.

6. 이미지 빌더에 업로드된 콘텐츠가 편집 내용과 일치하는지 확인하려면 panic 내용을 나열합니다.

```
# composer-cli blueprints show BLUEPRINT-NAME
```

7. Makefile 및 해당 종속 항목에 나열된 구성 요소 및 버전이 유효한지 확인합니다.

```
# composer-cli blueprints depsolve BLUEPRINT-NAME
```

## 4.4. 명령줄 인터페이스에서 이미지 빌더를 사용하여 시스템 이미지 생성

다음 절차에서는 Image Builder 명령줄 인터페이스를 사용하여 사용자 지정 이미지를 빌드하는 방법을 설명합니다.

**사전 요구 사항**

- 이미지에 대한 준비가 되어 있는 devfile이 있습니다.

**절차**

1. 작성을 시작합니다.

   ```
   # composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE
   ```

   *BLUEPRINT-NAME* 을 packaging의 이름으로 바꾸고 *IMAGE-TYPE* 을 이미지 유형으로 바꿉니다. 가능한 값은 **composer-cli compose 유형의** 출력을 참조하십시오.

   작성 프로세스는 백그라운드에서 시작되고 구성자 UUID(Universally Unique Identifier)를 보여줍니다.

2. 작성 프로세스가 완료될 때까지 기다립니다. 이미지 생성을 완료하는 데 최대 10분이 걸릴 수 있습니다.
   compose의 상태를 확인하려면 다음을 수행하십시오.

   ```
   # composer-cli compose status
   ```

   완료된 작성에는 상태 값인ANNISHED 가 표시됩니다. UUID를 통해 목록에서 compose를 식별합니다.

3. 작성 프로세스가 완료되면 결과 이미지 파일을 다운로드합니다.

   ```
   # composer-cli compose image UUID
   ```

   *UUID* 를 이전 단계에 표시된 UUID 값으로 바꿉니다.

**검증**

이미지를 생성한 후 다음 명령을 사용하여 이미지 생성 진행 상황을 확인할 수 있습니다.

- 작성 상태를 확인합니다.

  ```
  $ sudo composer-cli compose status
  ```

- 이미지의 메타데이터를 다운로드합니다.

  ```
  $ sudo composer-cli compose metadata UUID
  ```

- 이미지 로그를 다운로드합니다.

  ```
  $ sudo composer-cli compose logs UUID
  ```

명령은 이미지 생성 로그를 포함하는 **.tar** 파일을 생성합니다. 로그가 비어 있으면 저널을 확인할 수 있습니다.

- 저널을 확인합니다.

```
$ journalctl | grep osbuild
```

- 매니페스트를 확인합니다.

```
$ sudo cat /var/lib/osbuild-composer/jobs/job_UUID.json
```

저널에서 *job_UUID*.json을 찾을 수 있습니다.

**추가 리소스**

- 이미지 빌더 추적

## 4.5. 기본 이미지 빌더 명령줄 명령

Image Builder 명령줄 인터페이스는 다음 하위 명령을 제공합니다.

**RWO 조작**

**사용 가능한 모든 options 나열**

```
# composer-cli blueprints list
```

**TOML 형식의 RWO 콘텐츠 표시**

```
# composer-cli blueprints show BLUEPRINT-NAME
```

**TOML 형식의 devfile 콘텐츠를 파일BLUEPRINT-NAME.toml에 저장 (export)**

```
# composer-cli blueprints save BLUEPRINT-NAME
```

**제품 상세 정보**

```
# composer-cli blueprints delete BLUEPRINT-NAME
```

**TOML 형식의 crontab 파일을 이미지 빌더로 푸시(가져오기)**

```
# composer-cli blueprints push BLUEPRINT-NAME
```

**options에서 이미지 빌드**

**사용 가능한 이미지 유형 나열**

```
# composer-cli compose types
```

**작성을 시작**

```
# composer-cli compose start BLUEPRINT COMPOSE-TYPE
```

*BLUEPRINT* 를 빌드 및 *COMPOSE-TYPE* 을 출력 이미지 유형으로 바꿀 devfile의 이름으로 바꿉니다.

**모든 작성 항목 나열**

```
# composer-cli compose list
```

**모든 작성 및 상태 나열**

```
# composer-cli compose status
```

**실행 중인 작성 취소**

```
# composer-cli compose cancel COMPOSE-UUID
```

**완료된 작성을 삭제**

```
# composer-cli compose delete COMPOSE-UUID
```

**작성에 대한 자세한 정보 표시**

```
# composer-cli compose info COMPOSE-UUID
```

**작성 파일의 이미지 파일 다운로드**

```
# composer-cli compose image COMPOSE-UUID
```

**추가 리소스**

- *composer-cli*(1) 도움말 페이지는 사용 가능한 하위 명령 및 옵션의 전체 목록을 제공합니다.

  ```
  $ man composer-cli
  ```

- **composer-cli** 명령은 하위 명령 및 옵션에 대한 도움말을 제공합니다.

  ```
  # composer-cli help
  ```

## 4.6. 이미지 빌더 요건 형식

이미지 빌더 이메일은 TOML 형식의 일반 텍스트로 사용자에게 제공됩니다.

일반적인 options 파일의 요소는 다음과 같습니다.

**triggermes 메타데이터**

```
name = "BLUEPRINT-NAME"
description = "LONG FORM DESCRIPTION TEXT"
version = "VERSION"
```

*BLUEPRINT-NAME* 및 *LONG FORM DESCRIPTION TEXT* 를 devfile에 대한 이름 및 설명으로 바꿉니다.

*VERSION* 을 Semantic Versioning scheme에 따라 버전 번호로 바꿉니다.

이 부분은 전체 devfile 파일에 대해 한 번만 존재합니다.

entry *모듈*은 이미지에 설치할 패키지 이름 및 일치하는 버전 glob를 설명합니다.

항목 *그룹*은 이미지에 설치할 패키지 그룹을 설명합니다. 그룹은 패키지를 다음과 같이 분류합니다.

- 필수
- 기본값
- 선택 사항
  skopeo는 필수 패키지를 설치합니다. 선택적 패키지를 선택하는 메커니즘이 없습니다.

**이미지에 포함할 그룹**

```
[[groups]]
name = "group-name"
```

*group-name* 을 **anaconda-tools**, **widget**, **wheel** 또는 **users** 와 같은 그룹 이름으로 바꿉니다.

**이미지에 포함할 패키지**

```
[[packages]]
name = "package-name"
version = "package-version"
```

*package-name* 을 **httpd**, **gdb-doc** 또는 **coreutils** 와 같은 패키지 이름으로 바꿉니다.

사용할 *package-version* 을 버전으로 바꿉니다. 이 필드는 **dnf** 버전 사양을 지원합니다.

- 특정 버전의 경우 **8.30** 과 같은 정확한 버전 번호를 사용하십시오.
- 사용 가능한 최신 버전의 경우 별표 ***** 를 사용하십시오.
- 최신 마이너 버전의 경우 **8.***과 같은 형식을 사용하십시오.

포함할 모든 패키지에 대해 이 블록을 반복합니다.

## 4.7. 지원되는 이미지 사용자 지정

assembles에서는 여러 이미지 사용자 지정이 지원됩니다. 이러한 옵션을 사용하려면 처음에 탄력성에서 사용자 지정을 구성하고 이미지 빌더로 가져오기(push)해야 합니다.

**참고**

이러한 사용자 지정은 현재 웹 콘솔 내에서 지원되지 않습니다.

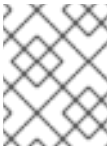
**이미지 호스트 이름 설정**

-

```
[customizations]
hostname = "baseimage"
```

**결과 시스템 이미지에 대한 사용자 사양**

```
[[customizations.user]]
name = "USER-NAME"
description = "USER-DESCRIPTION"
password = "PASSWORD-HASH"
key = "PUBLIC-SSH-KEY"
home = "/home/USER-NAME/"
shell = "/usr/bin/bash"
groups = ["users", "wheel"]
uid = NUMBER
gid = NUMBER
```



**참고**

GID는 선택 사항이며 이미지에 이미 있거나 패키지에 의해 생성되거나ceilometer **[[customizations.group]]** 항목에 의해 생성되어야 합니다.

**중요**

해시를 생성하려면 시스템에 **python3** 을 설치해야 합니다. 다음 명령은 **python3** 패키지를 설치합니다.

```
# yum install python3
```

managers *-HASH* 를 실제 암호 해시로 바꿉니다. 해시를 생성하려면 다음과 같은 명령을 사용합니다.

```
$ python3 -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
(pw==getpass.getpass("Confirm: ")) else exit())'
```

*PUBLIC-SSH-KEY* 를 실제 공개 키로 바꿉니다.

다른 자리 표시자를 적절한 값으로 바꿉니다.

필요에 따라 모든 행을 남겨 둡니다. 사용자 이름 만 필요합니다.

포함할 모든 사용자에 대해 이 블록을 반복합니다.

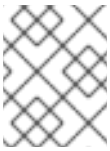
**결과 시스템 이미지에 대한 그룹 사양**

```
[[customizations.group]]
name = "GROUP-NAME"
gid = NUMBER
```

포함할 모든 그룹에 대해 이 블록을 반복합니다.

**기존 사용자 ssh 키 설정**

```
[[customizations.sshkey]]
user = "root"
key = "PUBLIC-SSH-KEY"
```



**참고**

이 옵션은 기존 사용자에게만 적용됩니다. 사용자를 생성하고 ssh 키를 설정하려면 이 섹션 **의 결과 시스템 이미지** 사용자 지정을 참조하십시오.

**커널 부팅 매개변수 옵션 추가**

```
[customizations.kernel]
append = "KERNEL-OPTION"
```

**기본적으로 이미지 빌더는 이미지에 기본 커널을 빌드합니다. 그러나 NetNamespace에서 다음과 같은 구성으로 커널을 사용자 지정할 수 있습니다.**

```
[customizations.kernel]
name = "KERNEL-rt"
```

**이미지에 사용할 커널 이름 정의**

```
[customizations.kernel.name]
name = "KERNEL-NAME"
```

**결과 시스템 이미지에 대한 시간대 및NTP(*Network Time Protocol*) 서버 설정**

```
[customizations.timezone]
timezone = "TIMEZONE"
ntpservers = "NTP_SERVER"
```

시간대를 설정하지 않으면 시스템은 *Universal Time, Coordinated* **(UTC)** 를 기본값으로 사용합니다. NTP 서버 설정은 선택 사항입니다.

**결과 시스템 이미지에 대한 로케일 설정 설정**

```
[customizations.locale]
languages = ["LANGUAGE"]
keyboard = "KEYBOARD"
```

언어 및 키보드 옵션을 모두 설정해야 합니다. 여러 언어를 추가할 수 있습니다. 추가하는 첫 번째 언어는 기본 언어이며 다른 언어는 보조 언어가 됩니다.

**결과 시스템 이미지에 대한 방화벽 설정**

```
[customizations.firewall]
port = ["PORTS"]
```

숫자 포트 또는 **/etc/services** 파일에서 해당 이름을 사용하여 목록을 활성화할 수 있습니다.

**방화벽 서비스 사용자 지정**

사용 가능한 방화벽 서비스를 검토합니다.

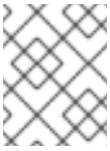
```
$ firewall-cmd --get-services
```

탄력성에서 사용자 **지정 섹션 사용자 지정.firewall.service** 는 사용자 지정하려는 방화벽 서비스를 지정합니다.

```
[customizations.firewall.services]
enabled = ["SERVICES"]
disabled = ["SERVICES"]
```

**firewall.services** 에 나열된 서비스는 **/etc/services** 파일에서 사용 가능한 이름과 다릅니다.

선택적으로 생성하려는 시스템 이미지에 대한 방화벽 서비스를 사용자 지정할 수 있습니다.



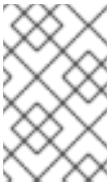
**참고**

방화벽 서비스를 사용자 정의하지 않으려면 Makefile에서 **[customizations.firewall]** 및 **[customizations.firewall.services]** 섹션을 생략합니다.

**부팅 시 활성화할 서비스 설정**

```
[customizations.services]
enabled = ["SERVICES"]
disabled = ["SERVICES"]
```

부팅 시 활성화할 서비스를 제어할 수 있습니다. 일부 이미지 유형에는 이미지가 올바르게 작동하고 이 설정을 재정의할 수 없도록 서비스가 이미 활성화되거나 비활성화되어 있습니다.



**참고**

빌드가 시작될 때마다 리포지토리를 복제합니다. 많은 양의 기록이 있는 리포지토리를 참조하는 경우 많은 양의 디스크 공간을 복제하고 사용하는 데 시간이 걸릴 수 있습니다. 또한 복제본은 일시적인 것이며 RPM 패키지가 생성된 후 제거됩니다.

**사용자 정의 파일 시스템 구성 지정**

새 디스크 레이아웃을 사용하여 사용자 지정 파일 시스템 구성을 지정하고 따라서 기본 레이아웃 구성을 사용하는 대신 특정 디스크 레이아웃을 사용하여 이미지를 생성할 수 있습니다. vGPU에 기본이 아닌 레이아웃 구성을 사용하면 다음과 같은 이점을 얻을 수 있습니다.

- 보안 벤치마크 준수
- 디스크 부족 오류 보호
- performance
- 기존 설정과의 일관성
  scaling에서 파일 시스템 구성을 사용자 정의합니다.

```
[[customizations.filesystem]]
mountpoint = "MOUNTPOINT"
size = MINIMUM-PARTITION-SIZE
```

다음 **마운트 지점** 및 하위 디렉터리가 지원됩니다.

- / - 루트 마운트 지점
- **/var**
- **/home**
- **/opt**
- **/srv**
- **/usr**
- **/app**
- **/data**

**참고**

마운트 지점 사용자 지정은 CLI를 사용하여 RHEL 8.5 및 RHEL 9.0 배포에서만 지원됩니다. 초기 배포에서는 **루트** 파티션만 마운트 지점으로 지정하고 **size** 인수를 이미지 크기의 별칭으로 지정할 수 있습니다.

둘 이상의 파티션이 있는 경우 LVM에 사용자 지정 파일 시스템 파티션으로 이미지를 생성하고 런타임 시 파티션의 크기를 조정할 수 있습니다. 이를 위해 청사진에서 사용자 지정 파일 시스템 구성을 지정한 다음 원하는 디스크 레이아웃을 사용하여 이미지를 생성할 수 있습니다. 기본 파일 시스템 레이아웃은 변경되지 않습니다. 파일 시스템 사용자 지정 없이 일반 이미지를 사용하면 **cloud-init** 에 의해 루트 파티션의 크기를 조정합니다.

청사진에 파일 시스템 사용자 지정을 추가한 후 파일 시스템이 LVM 파티션으로 변환됩니다.

*MINIMUM-PARTITION-SIZE* 를 정의할 때 기본 크기 형식이 없습니다. 다음 값과 단위가 지원됩니다. kB에서 TB로, KiB ~ TiB가 지원됩니다. 예를 들어 마운트 지점 크기를 바이트 단위로 정의할 수 있습니다.

```
[[customizations.filesystem]]
mountpoint = "/var"
size = 1073741824
```

단위를 사용하여 마운트 지점 크기를 정의할 수도 있습니다.

**참고**

RHEL 8.6 및 RHEL 9.0 배포에 제공된 패키지 버전의 단위를 사용하여 마운트 지점 크기를 정의할 수 있습니다.

예를 들면 다음과 같습니다.

```
[[customizations.filesystem]]
mountpoint = "/opt"
size = "20 GiB"
```

**추가 리소스**

- 파일 시스템 사용자 지정 "size"를 추가한 후 PKCS 가져오기가 실패합니다 .

## 4.8. 설치된 패키지

이미지 빌더를 사용하여 시스템 이미지를 생성할 때 기본적으로 시스템은 기본 패키지 세트를 설치합니다. 기본 패키지 목록은 **comps core** 그룹의 멤버입니다. 기본적으로 이미지 빌더는 **코어 yum** 그룹을 사용합니다.

**표 4.1. 이미지 유형 생성을 지원하는 기본 패키지**

| 이미지 유형 | 기본 패키지 |
|---|---|
| AMI | checkpolicy, chrony, cloud-utils, cloud-utils-growpart, @Core, dhcp-client, gdisk, insights-client, kernel, langpacks-en, net-tools, NetworkManager, redhat-release-eula, rng-tools, rsync, selinux-policy-targeted, yum-utils |
| openstack | @core, langpacks-en |
| qcow2 | @core, chrony, dnf, kernel, yum, nfs-utils, dnf-utils, cloud-init, python3-jsonschema, qemu-guest-agent, cloud-utils-growpart, dracut-norescue, tar, tcpdump, rsync, rsync, dnf-plugin-spacewalk, rhn-client-tools, rhnlib, rhn-setup, NetworkManager, dhcp-client, cockpit-ws, cockpit-system, subscription-manager-cockpit, redhat-release, redhat-release-eula, rng-tools, insights-client |
| tar | policycoreutils, selinux-policy-targeted |
| vhd | @core, langpacks-en |
| vmdk | @core, chrony, cloud-init, firewalld, langpacks-en, open-vm-tools, selinux-policy-targeted |

| 이미지 유형 | 기본 패키지 |
| --- | --- |
| edge-commit | attr, audit, basesystem, bash, bash, bash-completion, chrony, clevis-dracut, clevis-luks, container-selinux, coreutils,criu, cryptsetup, curl, dnsmasq, dosfstools, dracut-config-generic, dracut-network, e2fs, e2fsg, firewalld, fuse-overlayfs, fwupd, glibc, glibc-minimal-langpack, gnupg2, greenboot, gzip, hostname, ima-evm-utils, iproute, iptables, iputils, keyutils, less, lvm2, NetworkManager-wifi, NetworkManager-wwan, ns-altfiles, ns-altfiles OpenSSH-clients, openssh-server, passwd, pinentry, platform-python, platform-python, podman, policycoreutils, policycoreutils, polkit, procps-ng, redhat-release, rootfiles, rpm, rpm-ostree, rsync, selinux-policy-targeted, setools-console, setup, shadow-utils, setup Skopeo, slirp4netns, sudo, systemd, tar, tmux, traceroute, usbguard, util-linux, grep-minimal, wpa_supplicant, xz |
| edge-container | DNF, dosfstools, e2fsprogs, glibc, lorax-templates-rhel, lvm2, policycoreutils, python36, python3-iniparse, qemu-img, selinux-policy-targeted, systemd, tar, xfsprogs, xz |

| 이미지 유형 | 기본 패키지 |
| --- | --- |
| edge-installer | aajohan-comfortaa-fonts, abattis-cantarell-fonts, alsa-firmware, alsa-tools-firmware, anaconda, anaconda, anaconda-install-env-deps, anaconda-widgets, audit, bind-utils, bitmap-fangs, Fongfons controlPlane2, cryptsetup, dbus-x11, dejavu-sans-sans-mono-fonts, device-mapper-persistent-data, dnf, dump, ethtool, fcoe-utils, ftp, gdb-gdbserver, jenkinsfile, gdb-gdbserver, glibc-all-langs, glibc-all-langs Google-noto-sans-cjk-ttc-fonts, gsettings-Hellman-schemas, hdparm, hexedit, initscripts, ipmitool, iwl3945-firmware, iwl4965-firmware, iwl4965-firmware, iwl6000g2a-firmware, iwl6000grmware, iwl6000a-firmware Jomolhari-fonts, kacst-farsi-fonts, kacst-qurn-fonts, kbd, kbd-misc, kdump-anaconda-addon, khmeros-base-fonts, libblockdev-lvm-dbus, libblockdev-lvm-dbus, libertas-d86-d86 libertas-sd8787-firmware, libertas-usb8388-firmware, libertas-usb8388-olpc-firmware, libreport-plugin-bugzilla, libreport-plugin-bugzilla, libreport-plugin-reportuploader, libreport-rhel-anaconda-bugzilla librsvg2, linux-firmware, lklug-fonts, lldpad, lohit-assamese-fonts, lohit-bengali-fonts, lohit-devanagari-fonts, lohit-gujarati-fonts lohit-gurmukhi-fonts, lohit-kannada-fonts, lohit-odia-fonts, lohit-taung-fonts, lohit-telugu-fonts, lsof, madan-fonts, lsdan-fonts metacity, mt-st, mt-tools, nmap-ncat, nm-connection-editor, nss-connection-editor, openssh-server, oscap-anaconda-addon, pciutils, perlutils, perlutils, perl-interpreter, python3-pyatspi, rdma-core, redhat-release-esu, redhat-release-release rpm-ostree, rsync, rsyslog, sg3_utils, sil-abyssinica-fonts, sil-padauk-fonts, sil-scheherazade-fonts, smc-meera-fonts, smc-meera-fonts, spice-vdagent, strace, system-storage-manager, Thai-scalable-waree-fonts,kvmvnc-server-minimal, udisks2, udisks2-iscsi, usbutils, grep, volume_key, wget, xfsdump, xorg-x11-drivers,xorg-x11-fonts-misc, Xorg-x11-server-utils,xorg-x11-Xorg, xorg-x11-xauth |
| edge-simplified-installer | attr, basesystem, binutils, bsdtar, clevis-dracut, clevis-luks, cloud-utils-growpart, coreos-installer, coreos-installer-dracut, coreutils, device-mapper-multipath, dnsmasq, dosfstools, dracut-live, e2fsgs, e2fs FCoE-utils, fdo-init, gzip, ima-evm-utils, iproute, iputils, iscsi-initiator-utils, keyutils, lldpad, lvm2, passwd, policycoreutils, policycoreutils, policycoreutils-python-utils, procps-ng, rootfiles, setools-console, sudo, util-linux-linux |

| 이미지 유형 | 기본 패키지 |
| --- | --- |
| image-installer | Anaconda-dracut, curl, dracut-config-generic, dracut-network, hostname, iwl100-firmware, iwl1000-firmware, iwl105-firmware, iwl135-firmware, iwl2000-firmware, iwl2000-firmware, iwl203030-firmware, iwl2030-firmware iwl3160-firmware, iwl5000-firmware, iwl5150-firmware, iwl6000-firmware, iwl6050-firmware, iwl7260-firmware, kernel, less, nfs-utils, openssh-clients, ostree, ostree Plymouth, prefixdevname, rng-tools, rng-tools, selinux-policy-targeted, systemd, tar, xfsprogs, xz |
| edge-raw-image | DNF, dosfstools, e2fsprogs, glibc, lorax-templates-rhel, lvm2, policycoreutils, python36, python3-iniparse, qemu-img, selinux-policy-targeted, systemd, tar, xfsprogs, xz |

**참고**

배치에 구성 요소를 추가하는 경우 추가한 구성 요소의 패키지가 다른 패키지 구성 요소와 충돌하지 않는지 확인해야 합니다. 그렇지 않으면 시스템에서 종속성을 해결하지 못합니다. 결과적으로 사용자 지정된 이미지를 만들 수 없습니다.

**추가 리소스**

- 이미지 빌더 설명

## 4.9. 활성화된 서비스

사용자 지정 이미지를 구성하면 활성화된 서비스는 에서 **osbuild-composer** 를 실행 중인 RHEL 릴리스의 기본 서비스이며, 추가로 특정 이미지 유형에 활성화된 서비스가 활성화됩니다.

예를 들어 **.ami** 이미지 유형을 사용하면 서비스 **sshd**,**chronyd**, **cloud-init** 및 이러한 서비스가 없는 경우 사용자 지정 이미지가 부팅되지 않습니다.

**표 4.2. 이미지 유형 생성을 지원하는 활성화된 서비스**

| 이미지 유형 | 활성화된 서비스 |
| --- | --- |
| AMI | sshd, cloud-init, cloud-init-local, cloud-config, cloud-final |
| openstack | sshd, cloud-init, cloud-init-local, cloud-config, cloud-final |
| qcow2 | cloud-init |
| rhel-edge-commit | 추가 서비스는 기본적으로 활성화되지 않습니다. |

| 이미지 유형 | 활성화된 서비스 |
| --- | --- |
| tar | 추가 서비스는 기본적으로 활성화되지 않습니다. |
| vhd | sshd, chronyd, waagent, cloud-init, cloud-init-local, cloud-config, cloud-final |
| vmdk | sshd, chronyd, vmtoolsd, cloud-init |

참고: 시스템 부팅 중에 활성화할 서비스를 사용자 지정할 수 있습니다. 그러나 기본적으로 서비스가 활성화된 이미지 유형의 경우 사용자 지정은 이 기능을 재정의하지 않습니다.

### 추가 리소스

- 지원되는 이미지 사용자 지정

# 5장. 이미지 빌더 웹 콘솔 인터페이스를 사용하여 시스템 이미지 생성

이미지 빌더는 사용자 지정 시스템 이미지를 생성하기 위한 툴입니다. 이미지 빌더를 제어하고 사용자 지정 시스템 이미지를 만들려면 웹 콘솔 인터페이스를 사용할 수 있습니다. 명령줄 인터페이스는 더 많은 기능을 제공하므로 현재 권장되는 대안입니다.

## 5.1. RHEL 웹 콘솔에서 이미지 빌더 GUI에 액세스

RHEL 웹 콘솔의 **cockpit-composer** 플러그인을 사용하면 Image Builder 요건을 관리하고 그래픽 인터페이스를 사용하여 작성할 수 있습니다. Image Builder를 제어하는 기본 방법은 명령줄 인터페이스를 사용하는 현재입니다.

### 사전 요구 사항

- 시스템에 대한 루트 액세스 권한이 있어야 합니다.

### 절차

1. Image Builder가 설치된 시스템의 웹 브라우저에서 **https://localhost:9090/** 를 엽니다.
   Image Builder에 원격으로 액세스하는 방법에 대한 자세한 내용은 *RHEL 웹 콘솔 문서를 사용하여 시스템* 관리를 참조하십시오.
2. 시스템에 충분한 권한이 있는 사용자 계정의 자격 증명을 사용하여 웹 콘솔에 로그인합니다.
3. 이미지 빌더 컨트롤을 표시하려면 창의 왼쪽 상단에 있는 **Image Builder** 아이콘을 클릭합니다.
   Image Builder(이미지 빌더) 보기가 열리고 기존 options가 나열됩니다.

### 추가 리소스

- 이미지 빌더 명령줄 인터페이스를 사용하여 시스템 이미지 생성

## 5.2. 웹 콘솔 인터페이스에서 이미지 빌더 자격 증명 생성

사용자 지정 시스템 이미지를 설명하려면 먼저 Makefile을 생성합니다.

### 사전 요구 사항

- 브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열었습니다.

### 절차

1. 오른쪽 상단 **모서리에서 작업 생성** 을 클릭합니다.
   행동 이름 및 설명에 대한 필드가 포함된 팝업 창이 표시됩니다.
2. 행동의 이름, 설명을 입력한 다음 **생성** 을 클릭합니다.
   화면 변경(Selfific editing mode)입니다.
3. 시스템 이미지에 포함할 구성 요소를 추가합니다.
   a. 왼쪽에서 **Available Components** (사용 가능한 구성 요소) 필드에 구성 요소 이름의 전부 또는 일부를 입력하고 **Enter** 를 누릅니다.
   검색은 텍스트 항목 필드 아래의 필터 목록에 추가되고, 아래 구성 요소 목록은 검색과 일치하는 항목으로 줄어듭니다.

구성 요소 목록이 너무 길면 동일한 방식으로 검색 조건을 더 추가합니다.

b. 구성 요소 목록이 표시됩니다. 다른 결과 페이지로 이동하려면 구성 요소 목록의 화살표 및 입력 필드를 사용합니다.

c. 세부 정보를 표시하는 데 사용할 구성 요소의 이름을 클릭합니다. 오른쪽 창에는 버전 및 종속성과 같은 구성 요소의 세부 정보가 채워집니다.

d. 버전 **릴리스 드롭다운과 함께 구성 요소 옵션 상자에서 사용할 버전** 을 선택합니다.

e. 왼쪽 상단에 있는 **추가** 를 클릭합니다.

f. 구성 요소를 실수로 추가한 경우 **...을 클릭하여 제거하십시오.** 오른쪽 창에서 항목 맨 오른쪽에 있는 버튼을 클릭하고 메뉴에서 **제거를** 선택합니다.

**참고**

일부 구성 요소에 대한 버전을 선택하지 않으려면 구성 요소 목록 오른쪽에 있는 **+** 버튼을 클릭하여 구성 요소 세부 정보 화면 및 버전 선택을 건너뛸 수 있습니다.

4. capabilities를 저장하려면 오른쪽 상단에 있는 **Commit** (커밋)을 클릭합니다. 변경 사항이 요약되어 있는 대화 상자가 나타납니다. **Commit** 을 클릭합니다.
오른쪽의 작은 팝업은 저장 진행 상황을 안내 한 다음 결과를 알려줍니다.

5. 편집 화면을 종료하려면 왼쪽 상단에 있는 **찾기로 돌아가기** 를 클릭합니다.
Image Builder(이미지 빌더) 보기가 열리고 기존 options가 나열됩니다.

## 5.3. 웹 콘솔 인터페이스에서 이미지 빌더 에스컬레이션 편집

사용자 지정 시스템 이미지의 사양을 변경하려면 해당 Makefile을 편집합니다.

**사전 요구 사항**

- 브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열었습니다.
- 인테리어가 존재합니다.

**절차**

1. 왼쪽 상단에 있는 검색 상자에 이름 또는 해당 이름을 입력하고 **Enter** 를 눌러 편집할 Makefile을 찾습니다.
검색은 텍스트 항목 필드 아래의 필터 목록에 추가되고, 아래 options 목록은 검색과 일치하는 것으로 감소합니다.

   options 목록이 너무 길면 동일한 방식으로 검색 조건을 더 추가합니다.

2. Quarkus의 오른쪽에서 인식에 속한 **Editopenjdk** 버튼을 누릅니다.
options(스케줄) 편집 화면에 대한 보기 변경 사항.

3. 오른쪽 창에서 항목 맨 오른쪽에 있는 octets 버튼을 클릭하여 원하지 않는 구성 요소를 **제거하고 메뉴에서 제거를** 선택합니다.

4. 기존 구성 요소의 버전을 변경합니다.

   a. director 구성 요소 검색 필드에 구성 요소 이름 또는 구성 **요소** 부분을 **Enter** 키를 눌러 후

a. director 구성 요소 검색 필드에 구성 요소 이름 또는 구성 요소 부분을 **Enter** 키를 누른 후 Enter를 누릅니다.
검색은 텍스트 항목 필드 아래의 필터 목록에 추가되고, 아래 구성 요소 목록은 검색과 일치하는 항목으로 줄어듭니다.

구성 요소 목록이 너무 길면 동일한 방식으로 검색 조건을 더 추가합니다.

b. 구성 요소 항목의 맨 오른쪽에 있는 octets 버튼을 클릭하고 메뉴에서 **View** 를 선택합니다.

오른쪽 창에서 구성 요소 세부 정보 화면이 열립니다.

c. 버전 **릴리스 드롭다운 메뉴에서 원하는 버전** 을 선택하고 오른쪽 상단에 **변경 적용을** 클릭합니다.
변경 사항이 저장되고 오른쪽 창에는 credential 구성 요소가 나열되도록 돌아갑니다.

5. 새 구성 요소 추가:

a. 왼쪽에서 구성 요소 이름 또는 그 일부를 **Available Components** 제목 아래에 있는 필드에 입력하고 **Enter** 를 누릅니다.
검색은 텍스트 항목 필드 아래의 필터 목록에 추가되고, 아래 구성 요소 목록은 검색과 일치하는 항목으로 줄어듭니다.

구성 요소 목록이 너무 길면 동일한 방식으로 검색 조건을 더 추가합니다.

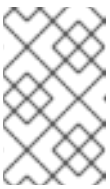
b. 구성 요소 목록이 표시됩니다. 다른 결과 페이지로 이동하려면 구성 요소 목록의 화살표 및 입력 필드를 사용합니다.

c. 세부 정보를 표시하는 데 사용할 구성 요소의 이름을 클릭합니다. 오른쪽 창에는 버전 및 종속성과 같은 구성 요소의 세부 정보가 채워집니다.

d. 버전 릴리스 드롭다운 메뉴에서 구성 요소 옵션 상자에서 사용할 버전을 선택합니다.Select the version you want to use in the **Component Options** box, with the **Version Release** drop-down menu.

e. 오른쪽 상단에 있는 **추가** 를 클릭합니다.

f. 구성 요소를 실수로 추가한 경우 오른쪽 창에 있는 항목의 맨 오른쪽에 있는 sandbox 버튼을 클릭하여 해당 구성 요소를 **제거한 후 메뉴에서 제거를** 선택합니다.



**참고**

일부 구성 요소에 대한 버전을 선택하지 않으려면 구성 요소 목록 오른쪽에 있는 **+** 버튼을 클릭하여 구성 요소 세부 정보 화면 및 버전 선택을 건너뛸 수 있습니다.

6. 변경 사항과 함께 새 버전의 containers를 커밋합니다.

a. 오른쪽 상단에 있는 **Commit** 버튼을 클릭합니다.
변경 사항에 대한 요약이 포함된 팝업 창이 표시됩니다.

b. 변경 사항을 검토하고 **커밋** 을 클릭하여 확인합니다.
오른쪽에 있는 작은 팝업은 저장 진행 상황 및 결과를 알려줍니다. 새로운 버전의 options이 생성되었습니다.

c. 왼쪽 상단에서 **Back to** introduces를 클릭하여 편집 화면을 종료합니다.
Image Builder(이미지 빌더) 보기가 열리고 기존 options가 나열됩니다.

## 5.4. 웹 콘솔 인터페이스의 이미지 빌더 DESKTOP에 사용자 및 그룹 추가

현재 웹 콘솔 인터페이스를 통해 사용자 및 그룹과 같은 사용자 지정을 사용할 수 없습니다. 이 제한 사항을 해결하려면 웹 콘솔의 **터미널** 탭을 사용하여 CLI(명령줄 인터페이스) 워크플로를 사용합니다.

**사전 요구 사항**

- 배란이 있어야 합니다.
- **vim**,**nano** 또는 **emacs** 와 같은 CLI 텍스트 편집기를 설치해야 합니다. 설치 방법:

```
# yum install editor-name
```

**절차**

1. Quarkus의 이름을 확인합니다. RHEL 웹 콘솔의 왼쪽에서**이미지 빌더**(이미지 빌더) 탭을 열어1) 이름을 확인합니다.
2. 웹 콘솔에서 CLI로 이동합니다. 왼쪽의 시스템 관리 탭을 열고 왼쪽 목록에서 마지막 항목 **Terminal** 을 선택합니다.
3. 슈퍼 사용자(root) 모드를 시작합니다.

```
$ sudo bash
```

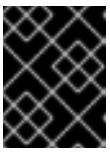
   묻는 메시지가 표시되면 자격 증명을 입력합니다. 웹 콘솔에 로그인할 때 터미널은 입력한 자격 증명을 재사용하지 않습니다.

   루트 권한이 있는 새 쉘은 홈 디렉터리에서 시작됩니다.

4. ceilometer를 파일로 내보냅니다.

```
# composer-cli blueprints save BLUEPRINT-NAME
```

5. 텍스트 편집기로 *BLUEPRINT-NAME*.toml 파일을 편집하고 사용자 및 그룹을 추가합니다.



**중요**

RHEL 웹 콘솔에는 시스템에서 텍스트 파일을 편집할 수 있는 기본 제공 기능이 없으므로 이 단계에서는 CLI 텍스트 편집기를 사용해야 합니다.

   a. 모든 사용자를 추가하려면 이 블록을 파일에 추가합니다.

```
[[customizations.user]]
name = "USER-NAME"
description = "USER-DESCRIPTION"
password = "PASSWORD-HASH"
key = "ssh-rsa (...) key-name"
home = "/home/USER-NAME/"
shell = "/usr/bin/bash"
groups = ["users", "wheel"]
uid = NUMBER
gid = NUMBER
```

managers -*HASH* 를 실제 암호 해시로 바꿉니다. 해시를 생성하려면 다음과 같은 명령을 사용합니다.

```
$ python3 -c 'import crypt,getpass;pw=getpass.getpass();print(crypt.crypt(pw) if
(pw==getpass.getpass("Confirm: ")) else exit())'
```

*ssh-rsa (...) key-name* 실제 공개 키가 있는 경우

다른 자리 표시자를 적절한 값으로 바꿉니다.

필요에 따라 모든 행을 남겨 둡니다. 사용자 이름 만 필요합니다.

b. 모든 사용자 그룹을 추가하려면 이 블록을 파일에 추가합니다.

```
[[customizations.group]]
name = "GROUP-NAME"
gid = NUMBER
```

c. 버전 번호를 늘립니다.

d. 파일을 저장하고 텍스트 편집기를 종료합니다.

6. packaging을 다시 이미지 빌더로 가져옵니다.

```
# composer-cli blueprints push BLUEPRINT-NAME.toml
```

**.toml** 확장자를 포함하여 파일 이름을 제공해야 하며, 다른 명령에서는 introduce의 이름만 사용해야 합니다.

7. 이미지 빌더에 업로드된 콘텐츠가 편집 내용과 일치하는지 확인하려면 panic 내용을 나열합니다.

```
# composer-cli blueprints show BLUEPRINT-NAME
```

버전이 파일에 넣은 항목과 일치하는지 확인하고 사용자 지정이 있는지 확인합니다.

**중요**

RHEL 웹 콘솔의 이미지 빌더 플러그인에는 devfile에 포함된 패키지를 편집하지 않는 한 변경 사항이 적용되었는지 확인하는 데 사용할 수 있는 정보가 표시되지 않습니다.

8. 권한 있는 쉘을 종료합니다.

```
# exit
```

9. 왼쪽에서 Image Builder(이미지**빌더**) 탭을 열고 모든 브라우저 및 열려 있는 모든 탭에서 페이지를 새로 고칩니다.
이렇게 하면 로드된 페이지에 캐시된 상태가 실수로 변경 사항을 되돌리지 않습니다.

**추가 리소스**

- 이미지 빌더 요건 형식
- 명령줄 인터페이스를 사용하여 이미지 빌더 승격 편집

## 5.5. 웹 콘솔 인터페이스에서 이미지 빌더를 사용하여 시스템 이미지 생성

다음 단계에서는 시스템 이미지 생성을 설명합니다.

**사전 요구 사항**

- 브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열었습니다.
- 인테리어가 존재합니다.

**절차**

1. 왼쪽 상단에 검색 상자에 이름 또는 일부를 입력하여 이미지를 빌드하려는 Makefile을 찾은 후 **Enter** 를 누릅니다.
   검색은 텍스트 항목 필드 아래의 필터 목록에 추가되고, 아래 options 목록은 검색과 일치하는 것으로 감소합니다.

   options 목록이 너무 길면 동일한 방식으로 검색 조건을 더 추가합니다.

2. Quarkus의 오른쪽에서 capabilities에 속하는 **이미지 생성** 버튼을 누릅니다.
   팝업 창이 나타납니다.

3. 이미지 유형을 선택하고 **Create** 를 누릅니다.
   오른쪽 상단에 있는 작은 팝업에서 이미지 생성이 큐에 추가되었음을 알려줍니다.

4. 프로비전 프로그램의 이름을 클릭합니다.
   options에 대한 세부 정보가 포함된 화면이 열립니다.

5. **이미지** 탭을 클릭하여 전환합니다. 생성 중인 이미지가 **진행** 중 상태로 나열됩니다.

   **참고**

   이미지 생성에 시간이 오래 걸리며 분 단위로 측정됩니다. 이미지가 생성되는 동안 진행 중은 표시되지 않습니다.

   이미지 생성을 중단하려면 오른쪽에서 **Stop** (중지) 버튼을 누릅니다.

6. 이미지가 성공적으로 생성되면 **중지** 버튼이 **다운로드** 버튼으로 교체됩니다. 이미지를 시스템에 다운로드하려면 이 버튼을 클릭합니다.

## 5.6. 프로비전에 소스 추가

이미지 빌더에 정의된 소스는 inventory에 추가할 수 있는 콘텐츠를 제공합니다. 이러한 소스는 전역적으로 사용되므로 모든 Makefile에서 사용할 수 있습니다. 시스템 소스는 컴퓨터에 로컬로 설정되며 이미지 빌더에서 제거할 수 없는 리포지토리입니다. 추가 사용자 지정 소스를 추가할 수 있으므로 시스템에서 사용 가능한 시스템 소스 이외의 다른 콘텐츠에 액세스할 수 있습니다.

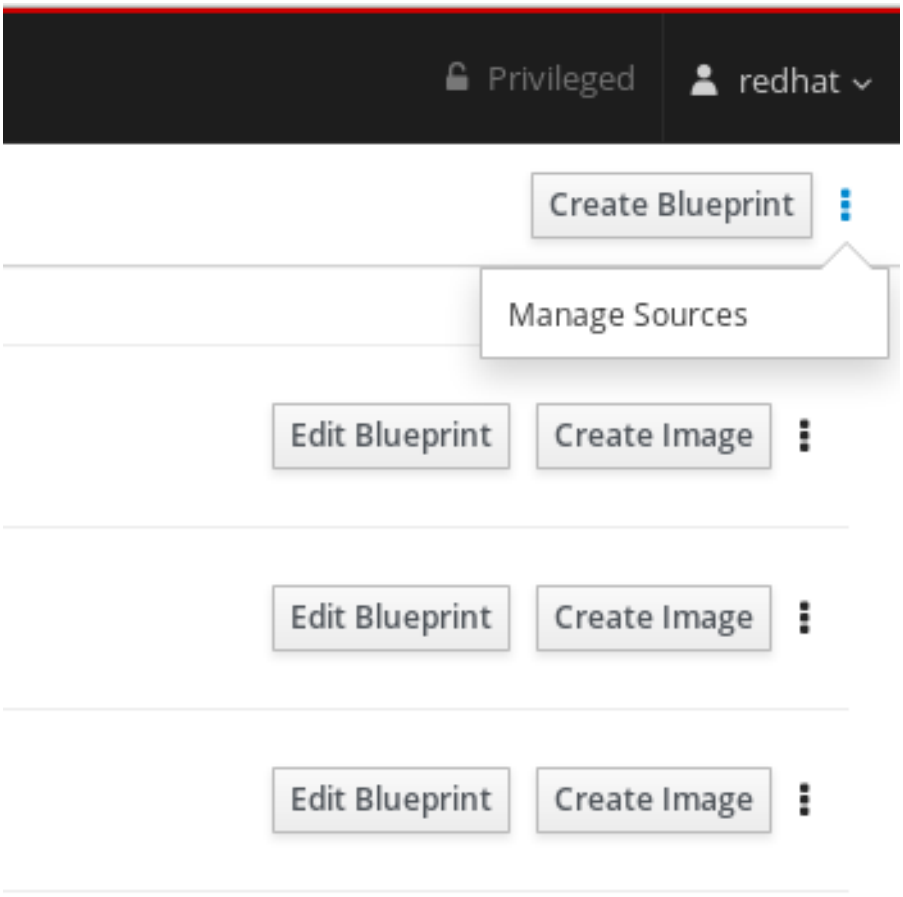
다음 단계에서는 로컬 시스템에 소스를 추가하는 방법을 설명합니다.

**사전 요구 사항**

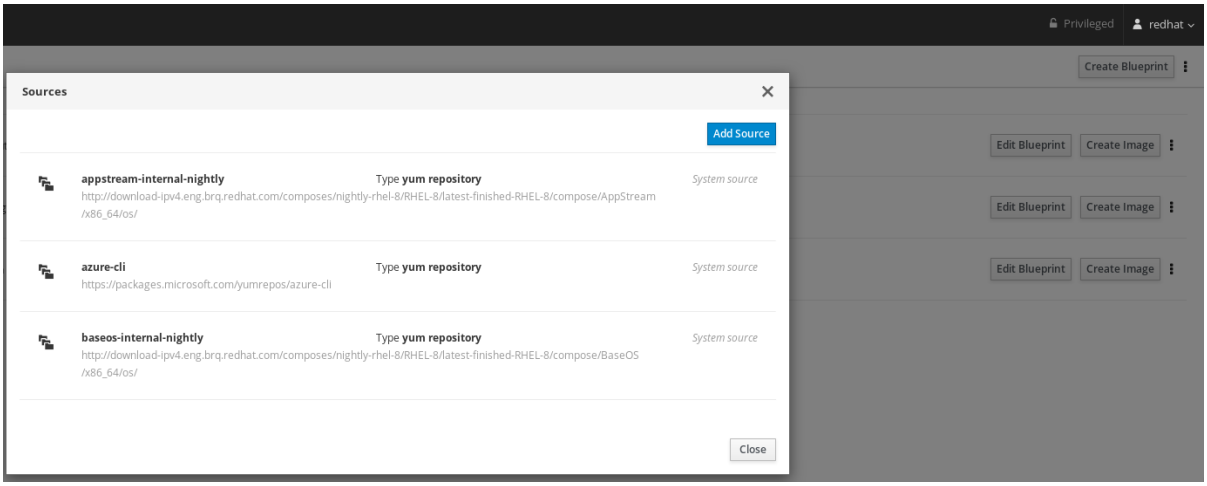
- 브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열었습니다.

**절차**

1. 오른쪽 상단에 있는 **소스 관리** 버튼을 클릭합니다.

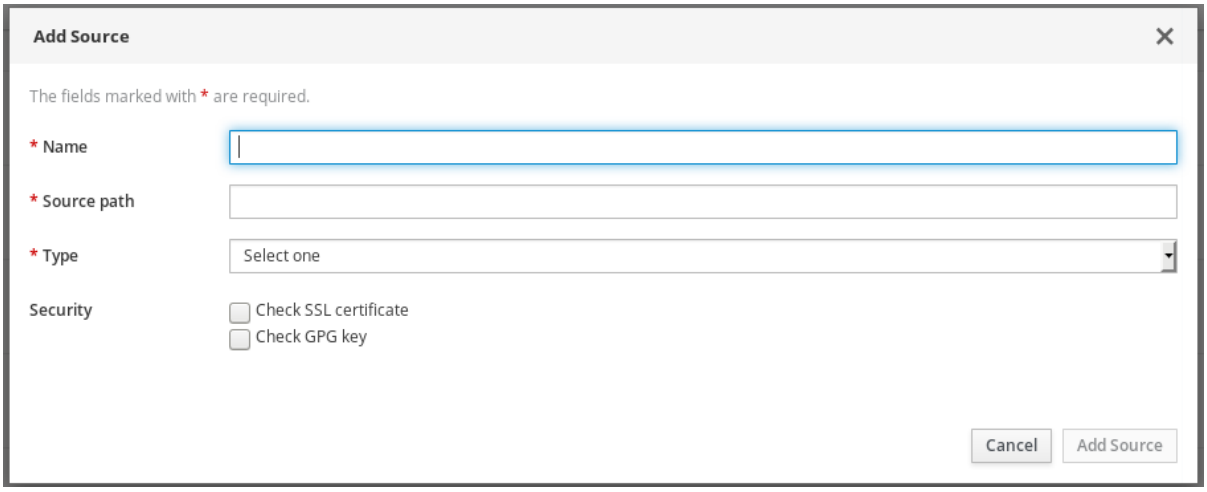



사용 가능한 소스, 이름 및 설명이 포함된 팝업 창이 표시됩니다.




2. 팝업 창 오른쪽에서 **소스 추가** 버튼을 클릭합니다.

3. 원하는 **소스 이름**, **소스 경로**, **소스** 유형을 추가합니다. **보안** 필드는 선택 사항입니다.




4. **소스 추가**를 클릭합니다. 화면에 사용 가능한 소스 창이 표시되고 추가한 소스 목록이 표시됩니다.

그 결과 새로운 시스템 소스를 사용할 수 있으며 사용 또는 편집할 준비가 되었습니다.

## 5.7. 프로비전에 대한 사용자 계정 생성

이미지 빌더에서 생성한 이미지에는 루트 계정이 잠겼고 다른 계정이 포함되어 있지 않습니다. 이러한 구성은 실수로 암호 없이 이미지를 빌드하고 배포할 수 없도록 제공됩니다. 이미지 빌더를 사용하면 credential에서 생성된 이미지에 로그인할 수 있도록 암호가 포함된 사용자 계정을 생성할 수 있습니다.

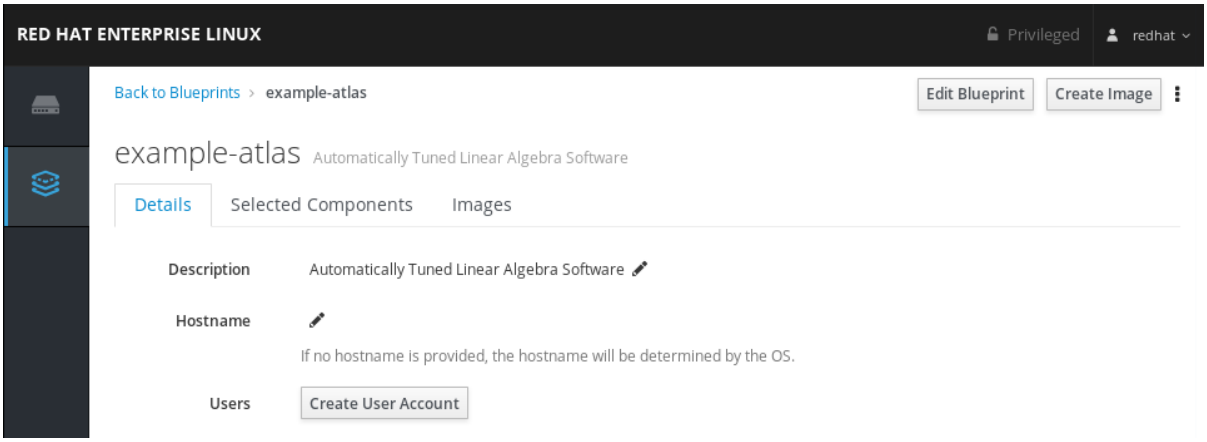
### 사전 요구 사항

- 브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열었습니다.
- 이미 존재하는 devfile이 있습니다.

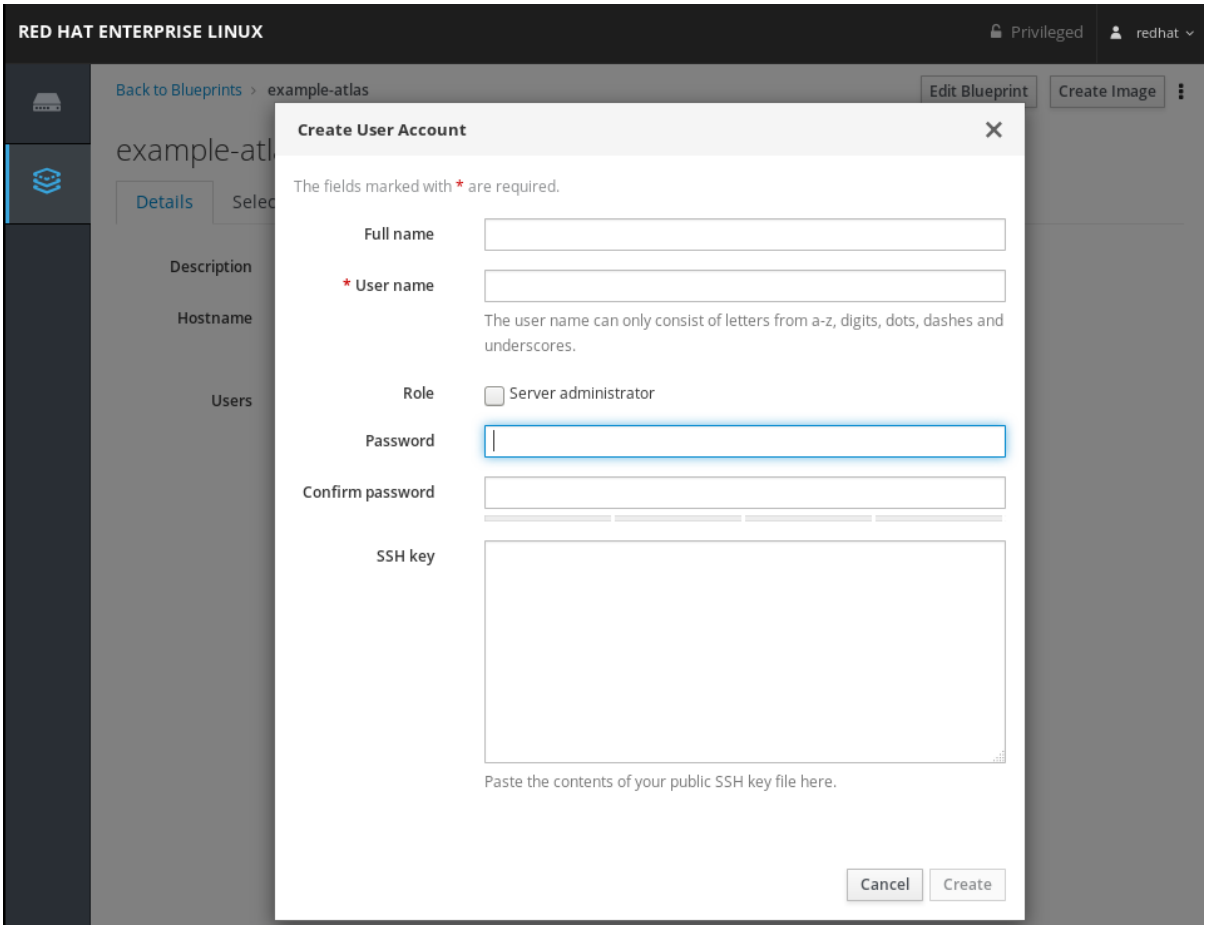
### 절차

1. 왼쪽 상단에 있는 검색 상자에 이름 또는 일부를 입력하여 사용자 계정을 생성하려는 Makefile을 찾은 후 **Enter** 를 누릅니다.
   검색은 텍스트 항목 필드 아래의 필터 목록에 추가되고, 아래 options 목록은 검색과 일치하는 것으로 감소합니다.

2. 프로비전 이름을 클릭하여 프로비전 세부 정보를 표시합니다.



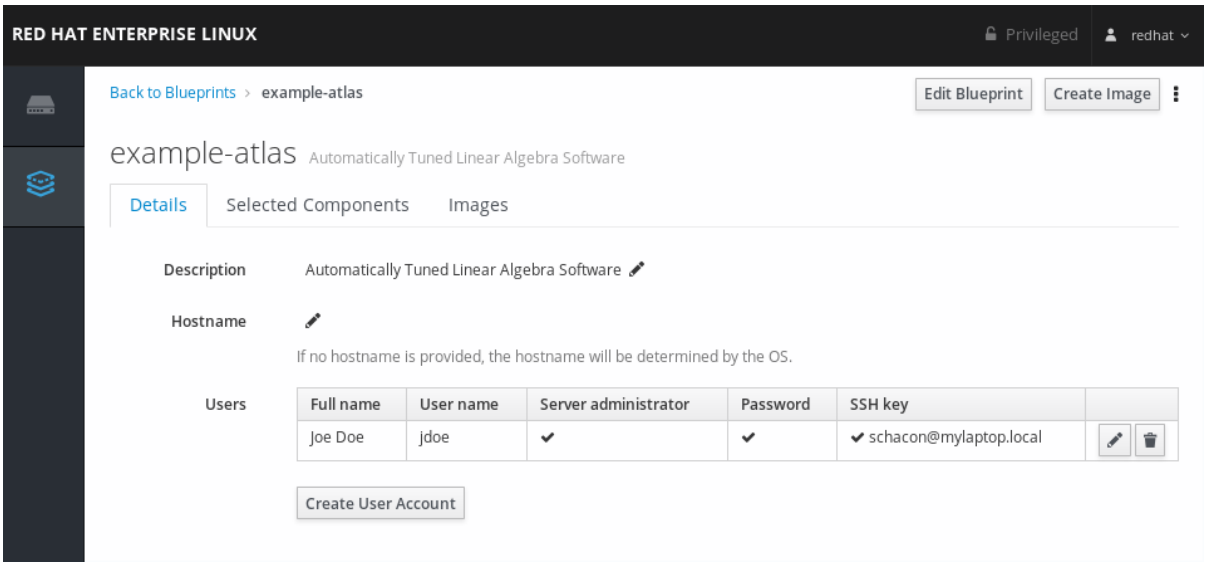

3. **Create User Account** 를 클릭합니다.
   그러면 사용자 계정 생성을 위한 필드가 포함된 창이 열립니다.




4. 세부 정보를 입력합니다. 이름을 삽입할 때 **사용자 이름** 필드를 자동 완성하여 사용자 이름을 제안합니다.

5. 원하는 세부 사항을 모두 삽입한 후 **생성을** 클릭합니다.

6. 삽입한 모든 정보를 보여주는 생성된 사용자 계정이 표시됩니다.




7. 프로비전에 대한 추가 사용자 계정을 생성하려면 프로세스를 반복합니다.

## 5.8. SSH 키를 사용하여 사용자 계정 생성

이미지 빌더에서 생성한 이미지에는 루트 계정이 잠겼고 다른 계정이 포함되어 있지 않습니다. 이러한 구성은 기본 암호가 없으므로 이미지의 보안을 유지하기 위해 제공됩니다. 이미지 빌더를 사용하면 devfile

에 대한 SSH 키를 사용하여 사용자 계정을 생성할 수 있으므로, 행동기에서 생성한 이미지에 인증할 수 있습니다. 이를 위해 먼저, 인위프리어를 만듭니다. 그런 다음 암호 및 SSH 키를 사용하여 사용자 계정을 만듭니다. 다음 예제에서는 SSH 키가 구성된 서버 관리자를 만드는 방법을 보여줍니다.

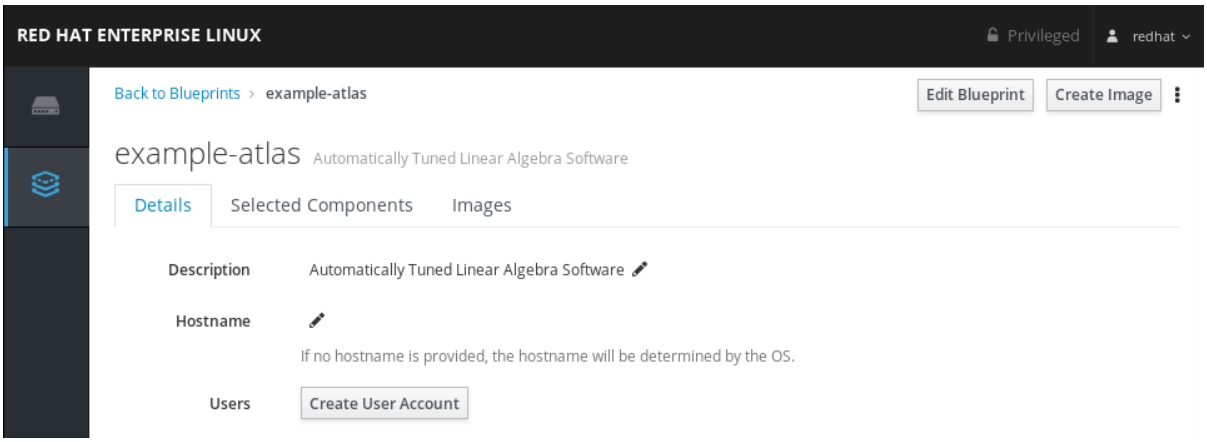
**사전 요구 사항**

- 생성된 사용자와 페어링될 SSH 키를 프로세스 후반부에서 생성했습니다.
- 브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열었습니다.
- 기존 credential을 보유하고 있습니다.

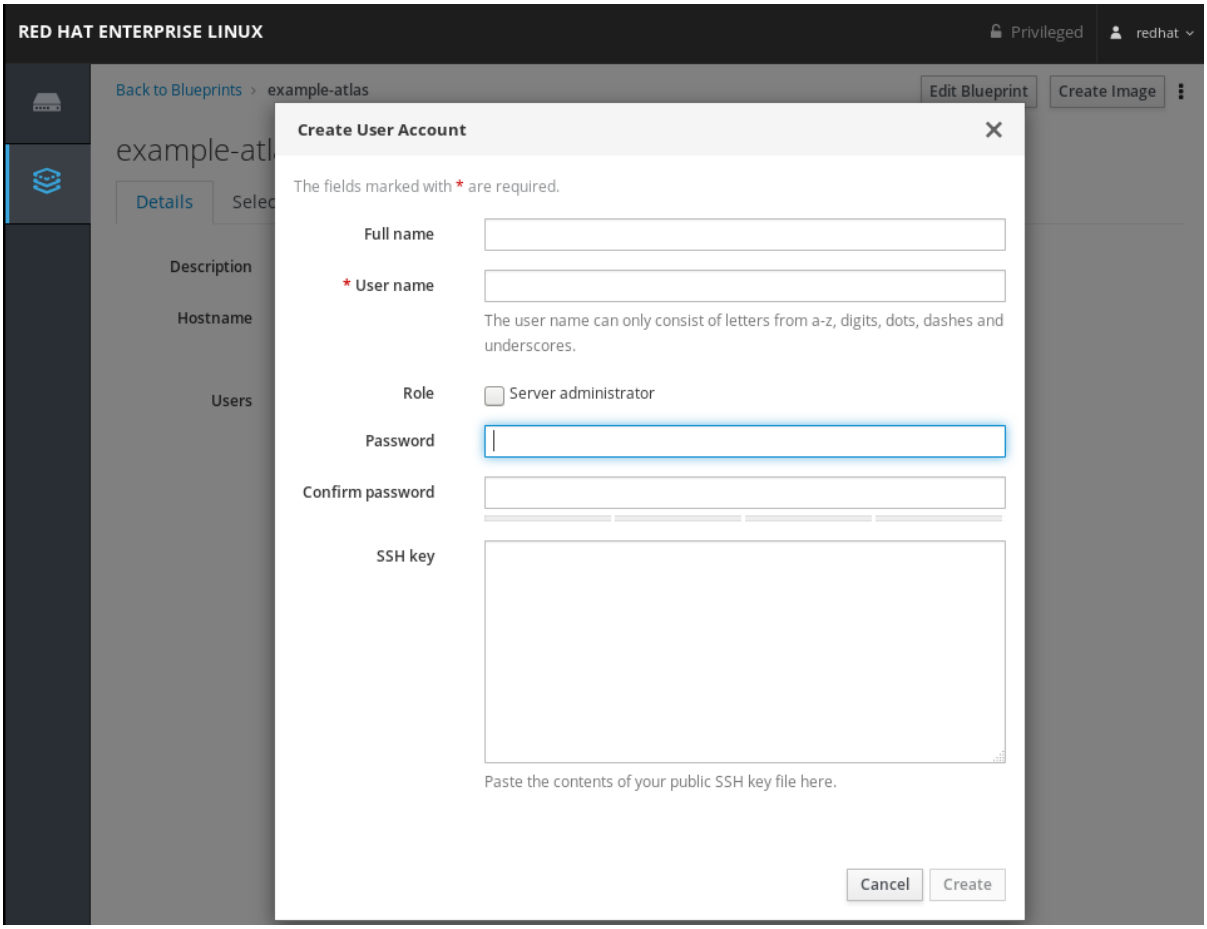
**절차**

1. 왼쪽 상단에 있는 검색 상자에 이름 또는 일부를 입력하여 사용자 계정을 생성하려는 Makefile을 찾은 후 **Enter** 를 누릅니다.
   검색은 텍스트 항목 필드 아래의 필터 목록에 추가되고, 아래 options 목록은 검색과 일치하는 것으로 감소합니다.

2. 프로비전 이름을 클릭하여 프로비전 세부 정보를 표시합니다.




3. **Create User Account** 를 클릭합니다.
   그러면 사용자 계정 생성을 위한 필드가 있는 창이 열립니다.

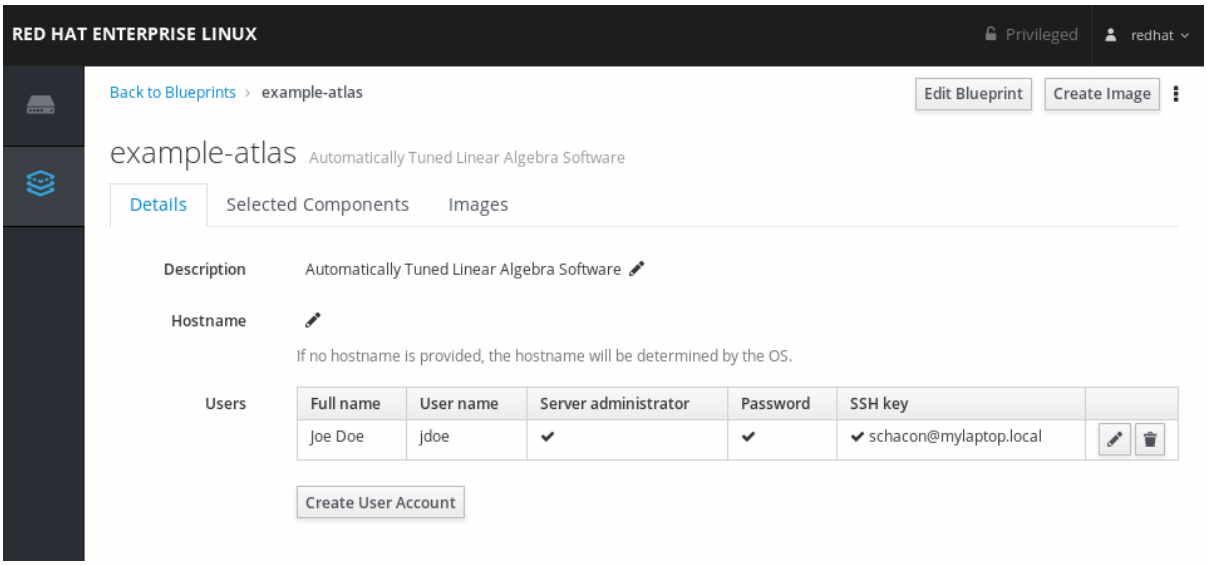



4. 세부 정보를 입력합니다. 이름을 삽입할 때 **사용자 이름** 필드를 자동 완성하여 사용자 이름을 제안합니다.
   생성 중인 사용자 계정에 관리자 권한을 제공하려면 **Role** 필드를 확인합니다.

   공개 SSH 키 파일의 내용을 붙여넣습니다.

5. 원하는 세부 사항을 모두 삽입한 후 **생성을** 클릭합니다.

6. 새 사용자 계정이 사용자 목록에 표시되어 삽입한 모든 정보가 표시됩니다.




7. 더 많은 사용자 계정을 생성하려는 경우 프로세스를 반복합니다.

**추가 리소스**

- SSH 키 쌍 생성

# 6장. 이미지 빌더를 사용하여 부팅 ISO 설치 프로그램 이미지 생성

Image Builder를 사용하여 부팅 가능한 ISO 설치 프로그램 이미지를 생성할 수 있습니다. 이러한 이미지는 루트 파일 시스템이 포함된 tarball으로 구성됩니다. 부팅 가능한 ISO 이미지를 사용하여 파일 시스템을 베어 메탈 서버에 설치할 수 있습니다.

Image Builder는 커밋과 루트 파일 시스템을 포함하는 부팅 ISO를 생성하는 매니페스트를 빌드합니다. ISO 이미지를 생성하려면 새 이미지 유형 **image-installer** 를 선택합니다. Image Builder는 다음을 포함하는 **.tar** 파일을 빌드합니다.

- 표준 Anaconda 설치 프로그램 ISO
- 임베디드 RHEL 시스템 tarball
- 최소 기본 요구 사항으로 커밋을 설치하는 기본 kickstart 파일

생성된 설치 프로그램 ISO 이미지는 베어 메탈 서버에 직접 설치할 수 있는 미리 구성된 시스템 이미지를 포함합니다.

## 6.1. 명령줄 인터페이스에서 이미지 빌더를 사용하여 부팅 ISO 설치 프로그램 이미지 생성

다음 절차에서는 Image Builder 명령줄 인터페이스를 사용하여 사용자 지정 부팅 ISO 설치 프로그램 이미지를 빌드하는 방법을 보여줍니다.

**사전 요구 사항**

- 사용자가 포함된 이미지의 청사진을 생성하고 Image Builder로 다시 푸시했습니다. 사용자의 블루프린트 사용자 지정을 참조하십시오.

**절차**

1. ISO 이미지를 생성합니다.

```
# composer-cli compose start BLUEPRINT-NAME image-installer
```

   - 생성한 블루프린트 이름이 있는 *BLUEPRINT-NAME*
   - *image-installer* 는 이미지 유형입니다.
     compose 프로세스는 백그라운드에서 시작되고 compose의 UUID가 표시됩니다.

2. 작성이 완료될 때까지 기다립니다. 이 작업은 몇 분 정도 걸릴 수 있습니다.
   compose의 상태를 확인하려면 다음을 수행하십시오.

```
# composer-cli compose status
```

   완료된 작성에는 상태 값이 FINISHED로 **표시됩니다**. UUID를 통해 목록에서 compose를 식별합니다.

3. 작성이 완료되면 결과 이미지 파일을 다운로드합니다.

```
# composer-cli compose image UUID
```

   *UUID* 를 이전 단계에 표시된 UUID 값으로 바꿉니다.

따라서 Image Builder는 ISO 설치 프로그램 이미지를 포함하는 **.tar** 파일을 빌드합니다.

**검증**

1. 이미지 파일을 다운로드한 폴더로 이동합니다.
2. 다운로드한 **.tar** 이미지를 찾습니다.
3. **.tar** 콘텐츠를 추출합니다.

생성된 ISO 이미지 파일을 하드 드라이브 또는 가상 시스템에서 부팅(예: HTTP 부팅 또는 USB 설치)할 수 있습니다.

**추가 리소스**

- 이미지 빌더 명령줄 인터페이스를 사용하여 시스템 이미지 생성
- RHEL의 부팅 가능 설치 미디어 생성

## 6.2. GUI에서 이미지 빌더를 사용하여 부팅 ISO 설치 프로그램 이미지 생성

이미지 빌더 GUI를 사용하여 사용자 지정된 부팅 ISO 설치 프로그램 이미지를 빌드할 수 있습니다.

**사전 요구 사항**

- 이미지에 대한akak을 생성했습니다.

**절차**

1. 브라우저에서 RHEL 웹 콘솔의 이미지 빌더 인터페이스를 엽니다.
2. 왼쪽 상단에 있는 검색 상자에 해당 이름 또는 일부를 입력하여 이미지를 빌드하려는 화면을 찾은 다음 **Enter** 를 클릭합니다.
3. 청사진의 오른쪽에 있는 Create Image(이미지 만들기) 버튼을 클릭하여 청사진에 속하는 **Create Image** (이미지 만들기) 버튼을 클릭합니다.
   **이미지 생성** 대화 마법사가 열립니다.
4. **이미지 유형 생성** 대화 상자의 **이미지 유형** 목록에서 다음을 수행합니다.
   a. **"RHEL Installer(.iso)"** 이미지 유형을 선택합니다.
   b. **생성**을 클릭합니다.

이미지 빌더는 RHEL **.iso** 이미지를 큐에 추가합니다.

**참고**

이미지 빌드 프로세스를 완료하는 데 몇 분이 걸립니다.

프로세스가 완료되면 이미지 빌드 완료 상태를 확인할 수 있습니다. 이미지 빌더는 **.iso** 이미지를 생성합니다.

**검증**

이미지가 성공적으로 생성되면 이미지 버튼을 다운로드할 수 있습니다.

1. **Download** 를 클릭하여 "RHEL Installer(.iso)" 이미지를 시스템에 저장합니다.
2. **RHEL 설치 프로그램(.iso)"** 이미지를 다운로드한 폴더로 이동합니다.
3. 다운로드한 .tar 이미지를 찾습니다.
4. **"RHEL Installer(.iso)"** 이미지 콘텐츠를 추출합니다.

생성된 ISO 이미지 파일을 하드 드라이브 또는 가상 시스템에서 부팅(예: HTTP 부팅 또는 USB 설치)할 수 있습니다.

**추가 리소스**

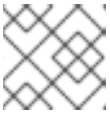
- 웹 콘솔 인터페이스에서 이미지 빌더 자격 증명 생성 .
- 이미지 빌더 명령줄 인터페이스를 사용하여 시스템 이미지 생성 .
- RHEL용 부팅 가능한 설치 매체 생성 .

## 6.3. 베어 메탈 시스템에 ISO 이미지 설치

다음 절차에서는 명령줄 인터페이스를 사용하여 이미지 빌더를 베어 메탈 시스템에 사용하여 생성한 부팅 가능한 ISO 이미지를 설치하는 방법을 설명합니다.

**사전 요구 사항**

- 이미지 빌더를 사용하여 부팅 가능한 ISO 이미지를 생성하셨습니다.
- 부팅 가능한 ISO 이미지를 다운로드 및 추출했습니다.
- 8GB USB 플래시 드라이브가 있습니다.



**참고**

ISO 크기는 인식기에서 선택한 패키지에 따라 더 커질 수 있습니다.

**절차**

1. 부팅 가능한 ISO 이미지 파일을 USB 플래시 드라이브에 배치합니다.
2. USB 플래시 드라이브를 부팅하려는 컴퓨터 포트에 연결합니다.
3. USB 플래시 드라이브에서 ISO 이미지를 부팅합니다.
4. 단계를 수행하여 사용자 지정된 부팅 가능한 ISO 이미지를 설치합니다.
   부팅 화면에는 다음 옵션이 표시됩니다.
   - Red Hat Enterprise Linux 8 설치
   - 이 미디어를 테스트하고 Red Hat Enterprise Linux 8 설치

**추가 리소스**

- 설치 부팅

# 7장. 이미지 빌더를 사용하여 KVM 게스트 이미지 준비 및 배포

고객은 ISO에서 수동으로 이미지를 생성하거나 이미지 빌더를 사용하여 생성한 용도의 이미지를 생성할 수 있습니다. 이 절차에서는 이미지 빌더를 사용하여 의도적으로 빌드된 이미지를 생성하는 단계를 설명합니다. 이는 RHV(Red Hat Virtualization)에 대한 **rhel-guest-image** 지원으로 제한됩니다.

다음과 같이 사용자 지정된 KVM 게스트 이미지를 생성하려면 다음과 같은 높은 수준의 단계가 포함됩니다.

1. 이미지 빌더를 사용하여 KVM 게스트 이미지 **.qcow2** 이미지 생성.
2. KVM 게스트 이미지에서 가상 머신 생성.

## 7.1. 이미지 빌더를 사용하여 사용자 지정 KVM 게스트 이미지 생성

이 명령은 이미지 빌더를 사용하여 **.qcow2** KVM 게스트 이미지를 생성하는 단계를 설명합니다.

**사전 요구 사항**

- 시스템에 root 또는 wheel 그룹 사용자가 액세스할 수 있어야 합니다.
- **cockpit-composer** 패키지가 설치되어 있습니다.
- RHEL 시스템에서 Cockpit UI의 이미지 빌더 대시보드를 열었습니다.

**절차**

1. **Createceilometer를** 클릭하여 프로비전을 생성합니다. 웹 콘솔 인터페이스에서 이미지 빌더 작업 생성을 참조하십시오.
2. 생성하는 KVM 게스트 이미지의 일부로 원하는 구성 요소 및 패키지를 선택합니다.
3. **Commit** (커밋)을 클릭하여 프로비전에 대한 변경 사항을 커밋합니다. 오른쪽의 작은 팝업은 저장 진행 상황을 안내 한 다음 커밋한 변경 사항의 결과를 알려줍니다.
4. 왼쪽 배너에서 요건 이름 링크를 클릭합니다.
5. **Images** (이미지) 탭을 선택합니다.
6. **Create Image** (이미지 만들기)를 클릭하여 사용자 지정 이미지를 만듭니다. 팝업 창이 열립니다.
7. **유형** 드롭다운 메뉴 목록에서 'QEMU Image(.qcow2)' 이미지를 선택합니다.
8. 이미지를 인스턴스화할 때 사용할 크기를 설정하고 **생성** 을 클릭합니다.
9. 창 오른쪽 상단에 있는 작은 팝업은 이미지 생성이 큐에 추가되었음을 알려줍니다. 이미지 생성 프로세스가 완료되면 **이미지 빌드 완료** 상태를 확인할 수 있습니다.

**검증 단계**

1. 이동 경로 아이콘을 클릭하고 **다운로드** 옵션을 선택합니다. 이미지 빌더는 기본 다운로드 위치에 KVM 게스트 이미지 **.qcow2** 파일을 다운로드합니다.

**추가 리소스**

- 웹 콘솔 인터페이스에서 이미지 빌더 자격 증명 생성 .

## 7.2. KVM 게스트 이미지에서 가상 머신 생성

이미지 빌더를 사용하면 **.qcow2** 이미지를 빌드하고 KVM 게스트 이미지를 사용하여 VM을 생성할 수 있습니다. 이미지 빌더를 사용하여 생성된 KVM 게스트 이미지에는 이미 **cloud-init** 가 설치되어 활성화되어 있습니다.

**사전 요구 사항**

- 이미지 빌더를 사용하여 **.qcow2** 이미지를 생성했습니다. 웹 콘솔 인터페이스에서 이미지 빌더 작업 생성을 참조하십시오.
- **qemu-kvm** 패키지가 시스템에 설치되어 있습니다. 시스템에서 **/dev/kvm** 폴더를 사용할 수 있는지 확인할 수 있습니다.
- 시스템에 **libvirt** 가 설치되어 있어야 합니다.
- 시스템에 **virt-install** 이 설치되어 있습니다.
- **genisoimage** 유틸리티가 시스템에 설치되어 있습니다.

**절차**

1. Image Builder를 사용하여 생성한 KVM 게스트 이미지를 **/var/lib/libvirt/images** 디렉터리로 이동하고 이미지 이름 이름을 **rhel-8.6-x86_64-kvm.qcow2** 로 변경합니다.
2. 예를 들어 **cloudinitiso** 디렉터리를 생성하고 새로 생성된 디렉터리로 이동합니다.

```
$ mkdir cloudinitiso
$ cd cloudinitiso
```

3. **meta-data** 라는 파일을 생성합니다. 이 파일에 다음 정보를 추가합니다.

```
instance-id: citest
local-hostname: vmname
```

4. **user-data** 라는 파일을 만듭니다. 파일에 다음 정보를 추가합니다.

```
#cloud-config
user: admin
password: password
chpasswd: {expire: False}
ssh_pwauth: True
ssh_authorized_keys:
  - ssh-rsa AAA...fhHQ== your.email@example.com
```

여기서,

- **ssh_authorized_keys** 는 SSH 공개 키입니다. SSH 공개 키는 **~/.ssh/id_rsa.pub** 에서 확인할 수 있습니다.

5. **genisoimage** 명령을 사용하여 **user-data** 및 **meta-data** 파일을 포함하는 ISO 이미지를 생성합니다.

```
# genisoimage -output cloud-init.iso -volid cidata -joliet -rock user-data meta-data

I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 331
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
183 extents written (0 MB)
```

6. **virt-install** 명령을 사용하여 KVM 게스트 이미지에서 새 VM을 생성합니다. 4단계에서 생성한 ISO 이미지를 VM 이미지에 첨부로 포함합니다.

```
# virt-install \
    --memory 4096 \
    --vcpus 4 \
    --name myvm \
    --disk rhel-8.6-x86_64-kvm.qcow2,device=disk,bus=virtio,format=qcow2 \
    --disk cloud-init.iso,device=cdrom \
    --os-variant rhel8.6 \
    --virt-type kvm \
    --graphics none \
    --import
```

여기서,

- --graphics none - 헤드리스 RHEL 8.4 VM입니다.
- --vCPUs 4 - 4 가상 CPU를 사용합니다.
- --memory 4096 -는 4096MB RAM을 사용함을 의미합니다.

7. VM 설치가 시작됩니다.

```
Starting install...
Connected to domain mytestcivm
...
[  OK  ] Started Execute cloud user/final scripts.
[  OK  ] Reached target Cloud-init target.

Red Hat Enterprise Linux 8.6 (Ootpa)
Kernel 4.18.0-221.el8.x86_64 on an x86_64
```

### 검증

부팅이 완료되면 VM에 텍스트 로그인 인터페이스가 표시됩니다. VM에 로그인하려면 다음을 수행합니다.

1. 사용자 이름으로 **admin** 을 입력하고 **Enter** 를 누릅니다.
2. 암호로 **password** 를 입력하고 **Enter** 를 누릅니다.
   로그인 인증이 완료되면 CLI를 사용하여 VM에 액세스할 수 있습니다.

# 8장. 이미지 빌더를 사용하여 클라우드 이미지 준비 및 업로드

**추가 resourcesit[cloud-init 중요한 디렉터리 및 파일].**

이미지 빌더는 다양한 공급자의 클라우드에서 사용할 수 있는 사용자 지정 시스템 이미지를 생성할 수 있습니다. 클라우드에서 사용자 지정된 RHEL 시스템 이미지를 사용하려면 해당 출력 유형을 사용하여 이미지 빌더로 시스템 이미지를 생성하고, 이미지를 업로드하도록 시스템을 구성하고, 이미지를 클라우드 계정에 업로드합니다. Red Hat Enterprise Linux 8.3에서 RHEL 웹 콘솔의 **Image Builder** 애플리케이션을 통해 사용자 정의 이미지 클라우드를 푸시하는 기능은 **AWS**, **Azure** 클라우드와 같은 서비스 공급자의 하위 집합에 사용할 수 있습니다. AWS Cloud AMI로 이미지 푸시 및 VHD 이미지를 Azure 클라우드로 푸시하는 것을 참조하십시오.

## 8.1. AWS AMI 이미지 업로드 준비

이는 AWS AMI 이미지를 업로드하기 위한 시스템을 구성하는 단계를 설명합니다.

**사전 요구 사항**

- AWS IAM 계정 관리자에 대한 액세스 키 ID가 구성되어 있어야 합니다.
- 쓰기 가능한 S3 버킷 이 준비되어 있어야 합니다.

**절차**

1. Python 3 및 **pip** 툴을 설치합니다.

```
# yum install python3
# yum install python3-pip
```

2. **pip** 를 사용하여 AWS 명령줄 툴을 설치합니다.

```
# pip3 install awscli
```

3. 다음 명령을 실행하여 프로필을 설정합니다. 터미널에 자격 증명, 지역 및 출력 형식을 입력하라는 메시지가 표시됩니다.

```
$ aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

4. 버킷 이름을 정의하고 다음 명령을 사용하여 버킷을 생성합니다.

```
$ BUCKET=bucketname
$ aws s3 mb s3://$BUCKET
```

*버킷* 이름을 실제 버킷 이름으로 교체합니다. 전역적으로 고유한 이름이어야 합니다. 그 결과 버킷이 생성됩니다.

5. 그런 다음 S3 버킷에 액세스할 수 있는 권한을 부여하려면 이전에 수행하지 않은 경우 IAM에서 **vmimport** S3 역할을 생성합니다.

```
$ printf '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service":
"vmie.amazonaws.com" }, "Action": "sts:AssumeRole", "Condition": { "StringEquals":{
"sts:Externalid": "vmimport" } } } ] }' > trust-policy.json
$ printf '{ "Version":"2012-10-17", "Statement":[ { "Effect":"Allow", "Action":[
"s3:GetBucketLocation", "s3:GetObject", "s3:ListBucket" ], "Resource":[ "arn:aws:s3:::%s",
"arn:aws:s3:::%s/*" ] }, { "Effect":"Allow", "Action":[ "ec2:ModifySnapshotAttribute",
"ec2:CopySnapshot", "ec2:RegisterImage", "ec2:Describe*" ], "Resource":"*" } ] }' $BUCKET
$BUCKET > role-policy.json
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-
policy.json
$ aws iam put-role-policy --role-name vmimport --policy-name vmimport --policy-document
file://role-policy.json
```

**추가 리소스**

- AWS CLI에서 상위 수준(s3) 명령 사용

## 8.2. CLI에서 AWS에 AMI 이미지 업로드

Image Builder를 사용하여 . **ami 이미지를** 빌드하고 CLI를 사용하여 Amazon AWS Cloud 서비스 공급자로 직접 푸시할 수 있습니다.

**사전 요구 사항**

- AWS IAM 계정 관리자에 대한 **액세스 키 ID** 가 구성되어 있습니다.
- 쓰기 가능한 S3 버킷 이 준비되었습니다.
- 정의된 options이 있습니다.

**절차**

1. 텍스트 편집기를 사용하여 다음 콘텐츠로 구성 파일을 생성합니다.

```
provider = "aws"

[settings]
accessKeyID = "AWS_ACCESS_KEY_ID"
secretAccessKey = "AWS_SECRET_ACCESS_KEY"
bucket = "AWS_BUCKET"
region = "AWS_REGION"
key = "IMAGE_KEY"
```

   필드의 값을 **accessKeyID**,**secretAccessKey**,**버킷**,**지역의** 자격 증명으로 바꿉니다.
   **IMAGE_KEY** 값은 EC2에 업로드할 VM 이미지의 이름입니다.

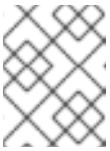
2. 파일을 *CONFIGURATION-FILE*.toml로 저장하고 텍스트 편집기를 종료합니다.

3. 작성을 시작합니다.

```
# composer-cli compose start BLUEPRINT-NAME IMAGE-TYPE IMAGE_KEY
CONFIGURATION-FILE.toml
```

   교체:

- *BLUEPRINT-NAME* 및 사용자가 생성한 Makefile의 이름으로
- **ami** 이미지 유형으로 *IMAGE-TYPE*.
- EC2에 업로드할 VM 이미지의 이름이 포함된 *IMAGE_KEY* 입니다.
- 클라우드 공급자의 구성 파일 이름으로 .toml를 구성합니다.



*참고*

*사용자 지정 이미지를 보낼 버킷에 대한 올바른 IAM 설정이 있어야 합니다. 이미지를 업로드하려면 먼저 버킷에 정책을 설정해야 합니다.*

4. *이미지 빌드 상태를 확인하고 AWS에 업로드합니다.*

```
# composer-cli compose status
```

*이미지 업로드 프로세스가 완료되면 "FINISHED" 상태를 확인할 수 있습니다.*

*검증*

*이미지가 업로드되었는지 확인하려면 다음을 수행하십시오.*

1. *메뉴에서 EC2 에 액세스하고 AWS 콘솔에서 올바른 리전을 선택합니다. 이미지가 업로드되었음을 나타내려면 이미지에 "사용 가능" 상태가 있어야 합니다.*
2. *대시보드에서 이미지를 선택하고 **시작을** 클릭합니다.*

*추가 리소스*

- *VM을 가져오는 데 필요한 서비스 역할*

## *8.3. AWS CLOUD AMI로 이미지 푸시*

*이번에는 **AWS Cloud AMI** 에 생성한 출력 이미지를 푸시할 수 있습니다. 여기서는 Image Builder를 사용하여 생성한 **.ami** 이미지를 Amazon AWS Cloud 서비스 공급자로 내보내는 단계를 설명합니다.*

*사전 요구 사항*

- *시스템에 **root** 또는 **wheel** 그룹 사용자가 액세스할 수 있어야 합니다.*
- *브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열었습니다.*
- *AWS IAM 계정 관리자에 대한 액세스 키 ID가 구성되어 있어야 합니다.*
- *쓰기 가능한 S3 버킷 이 준비되어 있어야 합니다.*

*절차*

1. ***Createceilometer를** 클릭하여 프로비전을 생성합니다. 웹 콘솔 인터페이스에서 이미지 빌더 작업 생성을 참조하십시오.*
2. *생성 중인 이미지의 일부로 원하는 구성 요소 및 패키지를 선택합니다.*
3. ***Commit** (커밋)을 클릭하여 프로비전에 대한 변경 사항을 커밋합니다.*

오른쪽의 작은 팝업은 저장 진행 상황을 안내 한 다음 커밋한 변경 사항의 결과를 알려줍니다.

4. 왼쪽 배너에서 bin **name** 링크를 클릭합니다.

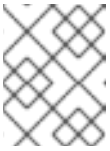
5. ***Images*** (이미지) 탭을 선택합니다.

6. ***Create Image*** (이미지 만들기)를 클릭하여 사용자 지정 이미지를 만듭니다.
팝업 창이 열립니다.

   a. ***"Type"*** 드롭다운 메뉴 목록에서 'Amazon Machine Image Disk (.ami)"' 이미지를 선택합니다.

   b. ***"AWS로 업로드"*** 확인란을 선택하여 이미지를AWS 클라우드에 업로드하고 ***다음을*** 클릭합니다.

   c. AWS에 대한 액세스를 인증하려면 해당 필드에 "AWS 액세스 키 ID" 및 "AWS 보안 액세스 키"를 입력합니다. ***다음***을 클릭합니다.



***참고***

새 액세스 키 ID를 생성하는 경우에만AWS 시크릿 액세스 키를 볼 수 있습니다. 보안 키를 모르는 경우 새 액세스 키 ID를 생성합니다.

   d. "Image name" 필드에 이미지 이름을 입력하고 "Amazon S3 버킷 이름" 필드에 Amazon 버킷 이름을 입력하고 사용자 지정된 이미지를 추가할 버킷에 대해 "AWS region" 필드를 입력합니다. ***다음***을 클릭합니다.

   e. 정보를 검토하고 ***완료*** 를 클릭합니다.
경우에 따라 ***뒤로*** 클릭하여 잘못된 세부 정보를 수정할 수 있습니다.

***참고***

사용자 지정 이미지를 보낼 버킷에 대한 올바른IAM 설정이 있어야 합니다. IAM 가져오기 및 내보내기를 사용하므로 이미지를 업로드하기 전에 버킷에 ***정책을*** 설정해야 합니다. 자세한 내용은 *IAM 사용자에 대한 필수 권한 을* 참조하십시오.

7. 오른쪽의 작은 팝업은 저장 진행 상황을 알려줍니다. 또한 이미지 생성, 이 이미지 생성 진행 및 후속 업로드AWS 클라우드에 대한 이미지 생성이 시작되었음을 알려줍니다.
프로세스가 완료되면 ***"이미지 빌드 완료"*** 상태를 확인할 수 있습니다.

8. 메뉴에서 ServiceoctetsEC2 를 클릭하고AWS 콘솔에서 *올바른 리전* 을 선택합니다. 이미지가 업로드되었음을 나타내려면 이미지에 "사용 가능" 상태가 있어야 합니다.

9. 대시보드에서 이미지를 선택하고 ***시작을*** 클릭합니다.

10. 새 창이 열립니다. 이미지를 시작하는 데 필요한 리소스에 따라 인스턴스 유형을 선택합니다. ***검토 및 시작을*** 클릭합니다.

11. 인스턴스 시작 세부 정보를 검토합니다. 변경을 수행해야 하는 경우 각 섹션을 편집할 수 있습니다. ***시작을 클릭합니다.***

12. 인스턴스를 시작하기 전에 액세스할 공개 키를 선택해야 합니다.
이미 보유하고 있는 키 쌍을 사용하거나 새 키 쌍을 만들 수 있습니다. 또는 ***이미지 빌더*** 를 사용하여 사전 정의된 공개 키가 있는 이미지에 사용자를 추가할 수 있습니다. 자세한 내용은SSH 키를 *사용하여 사용자 계정 생성* 을 참조하십시오.

다음 단계에 따라 EC2에서 새 키 쌍을 생성하고 새 인스턴스에 연결합니다.

a. 드롭다운 메뉴 목록에서 **"Create a new key pair"**을 선택합니다.

b. 새 키 쌍의 이름을 입력합니다. 새 키 쌍을 생성합니다.

c. **"Download Key pair"** 을 클릭하여 로컬 시스템에 새 키 쌍을 저장합니다.

13. 그런 다음 **Launch Instance** (인스턴스 시작)를 클릭하여 인스턴스를 시작할 수 있습니다. 인스턴스의 상태를 확인할 수 있으며, **"Initializing"** 으로 표시됩니다.

14. 인스턴스 상태가 **"실행 중"** 이면 **Connect** 버튼을 사용할 수 있게 됩니다.

15. **연결**을 클릭합니다. SSH를 사용하여 연결하는 방법에 대한 지침이 포함된 팝업 창이 표시됩니다.

a. **"독립 실행형 SSH 클라이언트"** 에 대한 기본 연결 방법을 선택하고 터미널을 엽니다.

b. 개인 키를 저장한 위치에서 SSH가 작동하도록 키를 공개적으로 볼 수 있는지 확인합니다. 이렇게 하려면 명령을 실행합니다.

```
$ chmod 400 <your-instance-name.pem>_
```

c. 공용 DNS를 사용하여 인스턴스에 연결합니다.

```
$ ssh -i "<_your-instance-name.pem_"> ec2-user@<_your-instance-IP-address_>
```

d. "yes"를 입력하여 연결을 계속할지 확인합니다.
결과적으로 SSH를 사용하여 인스턴스에 연결합니다.

**검증 단계**

1. SSH를 사용하여 인스턴스에 연결된 상태에서 모든 작업을 수행할 수 있는지 확인합니다.

**추가 리소스**

- Red Hat 고객 포털에서 케이스 만들기
- SSH를 사용하여 Linux 인스턴스에 연결
- 지원 케이스 공개

## 8.4. AZURE VHD 이미지 업로드 준비

이에서는 Azure에 VHD 이미지를 업로드하는 단계를 설명합니다.

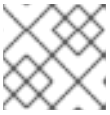
**사전 요구 사항**

- 사용 가능한 Azure 리소스 그룹 및 스토리지 계정이 있어야 합니다.

**절차**

1. python2를 설치합니다.

```
# yum install python2
```



**참고**

*AZ CLI는 특히 python 2.7에 의존하기 때문에 **python2** 패키지를 설치해야 합니다.*

2. *Microsoft 리포지토리 키를 가져옵니다.*

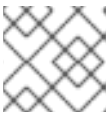
```
# rpm --import https://packages.microsoft.com/keys/microsoft.asc
```

3. *로컬 azure-cli 리포지토리 정보를 생성합니다.*

```
# sh -c 'echo -e "[azure-cli]\nname=Azure
CLI\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
cli\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" >
/etc/yum.repos.d/azure-cli.repo'
```

4. *Azure CLI를 설치합니다.*

```
# yumdownloader azure-cli
# rpm -ivh --nodeps azure-cli-2.0.64-1.el7.x86_64.rpm
```



**참고**

*다운로드한 Azure CLI 패키지는 현재 다운로드한 버전에 따라 다를 수 있습니다.*

5. *Azure CLI를 실행합니다.*

```
$ az login
```

*터미널에 '참고, 로그인할 수 있는 브라우저를 시작 했습니다. 장치 코드에 대한 오래된 경험의 경우 "az login --use-device-code"를 사용하고 로그인할 수 있는 브라우저를 엽니다.*

**참고**

*원격(SSH) 세션을 실행 중인 경우 브라우저에 링크가 표시되지 않습니다. 이 경우 제공된 링크를 사용하여 원격 세션에 로그인하고 인증할 수 있습니다. 로그인하려면 웹 브라우저를 사용하여 https://microsoft.com/devicelogin 페이지를 열고 인증할 코드 XXXReplicas를 입력합니다.*

6. *Azure의 스토리지 계정 키를 나열합니다.*

```
$ GROUP=resource-group-name
$ ACCOUNT=storage-account-name
$ az storage account keys list --resource-group $GROUP --account-name $ACCOUNT
```

*resource-group-name 을 Azure 리소스 그룹 이름으로 바꾸고 storage-account-name 을 Azure 스토리지 계정 이름으로 교체합니다.*

**참고**

*명령을 사용하여 사용 가능한 리소스를 나열할 수 있습니다.*

```
$ az resource list
```

7. 이전 명령의 출력에서 value **key1** 을 적어 두고 환경 변수에 할당합니다.

```
$ KEY1=value
```

8. 스토리지 컨테이너를 생성합니다.

```
$ CONTAINER=storage-account-name
$ az storage container create --account-name $ACCOUNT \
--account-key $KEY1 --name $CONTAINER
```

storage-account-name 을 스토리지 계정 이름으로 교체합니다.

### 추가 리소스

- Azure CLI

## 8.5. AZURE에 VHD 이미지 업로드

이에서는Azure에 VHD 이미지를 업로드하는 단계를 설명합니다.

### 사전 요구 사항

- Azure VHD 이미지를 업로드하도록 시스템을 설정해야 합니다.
- 이미지 빌더에서 생성한Azure VHD 이미지가 있어야 합니다. 이미지를 생성할 때 GUI에서 CLI 또는 **Azure Disk Image (.vhd )** 에서 vhd 출력 유형을 사용합니다.

### 절차

1. 이미지를Azure로 푸시하고 해당 이미지에서 인스턴스를 생성합니다.

```
$ VHD=25ccb8dd-3872-477f-9e3d-c2970cd4bbaf-disk.vhd
$ az storage blob upload --account-name $ACCOUNT --container-name $CONTAINER -
-file $VHD --name $VHD --type page
...
```

2. Azure BLOB에 업로드가 완료된 후Azure 이미지를 만듭니다.

```
$ az image create --resource-group $GROUP --name $VHD --os-type linux --location
eastus --source https://$ACCOUNT.blob.core.windows.net/$CONTAINER/$VHD
 - Running ...
```

3. Azure Portal을 사용하여 인스턴스를 생성하거나 다음과 유사한 명령을 생성합니다.

```
$ az vm create --resource-group $GROUP --location eastus --name $VHD --image
$VHD --admin-username azure-user --generate-ssh-keys
 - Running ...
```

4. SSH를 통해 개인 키를 사용하여 결과 인스턴스에 액세스합니다. **azure-user** 로 로그인합니다.

*추가 리소스*

- ***.vhd** 형식의 이미지 구성 실패*

## 8.6. VSPHERE 에 VMDK 이미지 업로드

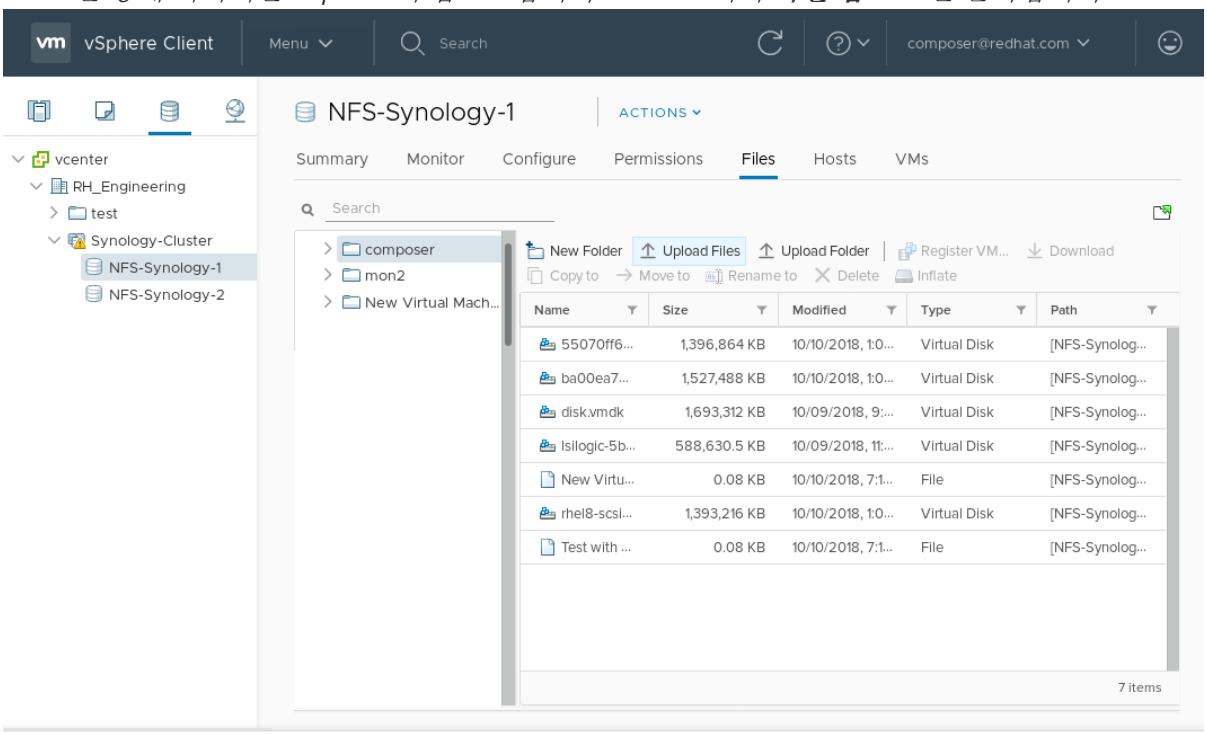
*이미지 빌더는 VMware ESXi 또는 vSphere 시스템에 업로드하는 데 적합한 이미지를 생성할 수 있습니다. 이 명령은 VMDK 이미지를 VMware vSphere 에 업로드하는 단계를 설명합니다.*

*사전 요구 사항*

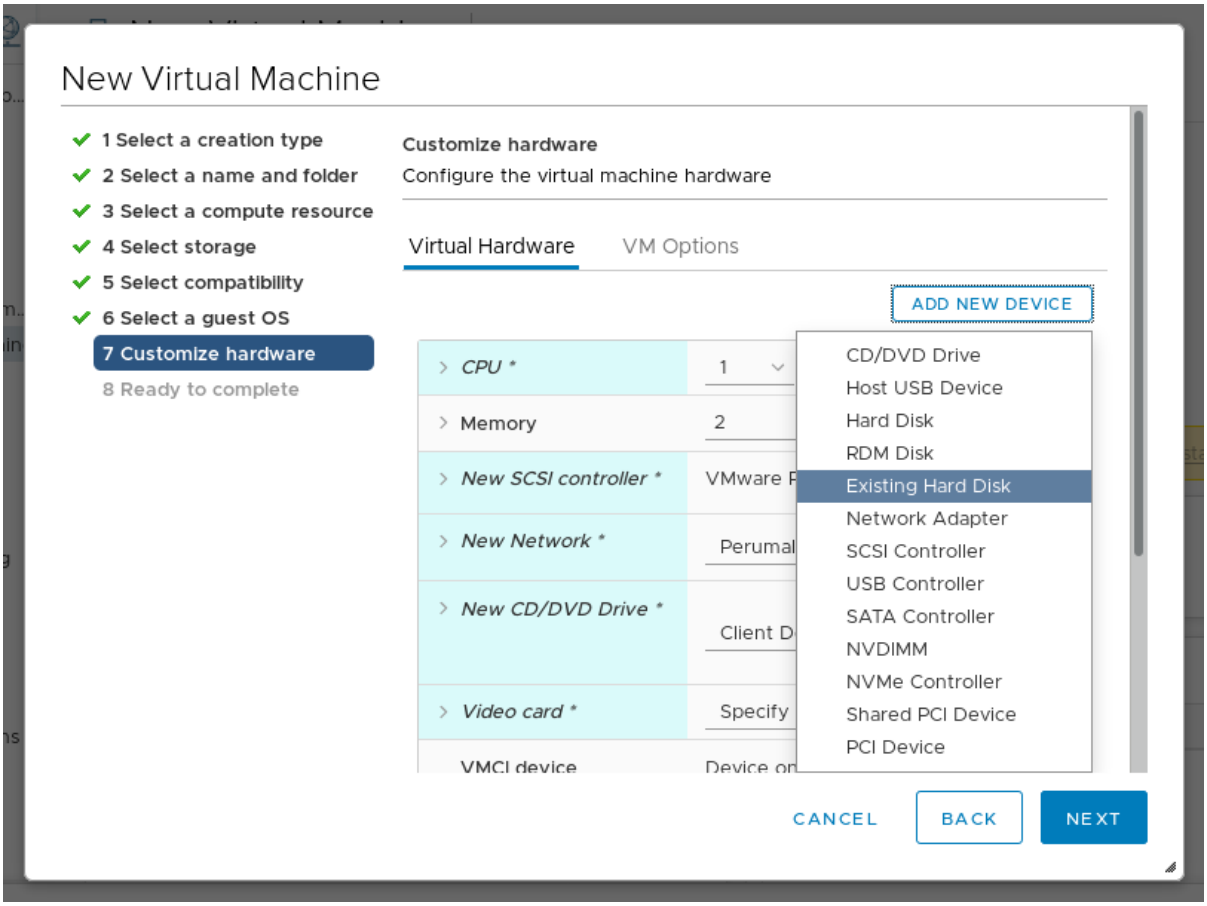
- *이미지 빌더에서 생성한 VMDK 이미지가 있어야 합니다. 이미지를 생성할 때 GUI 에서 CLI 또는 **VMware Virtual Machine Disk(.vmdk)** 에서 vmdk 출력 유형을 사용합니다.*

*절차*

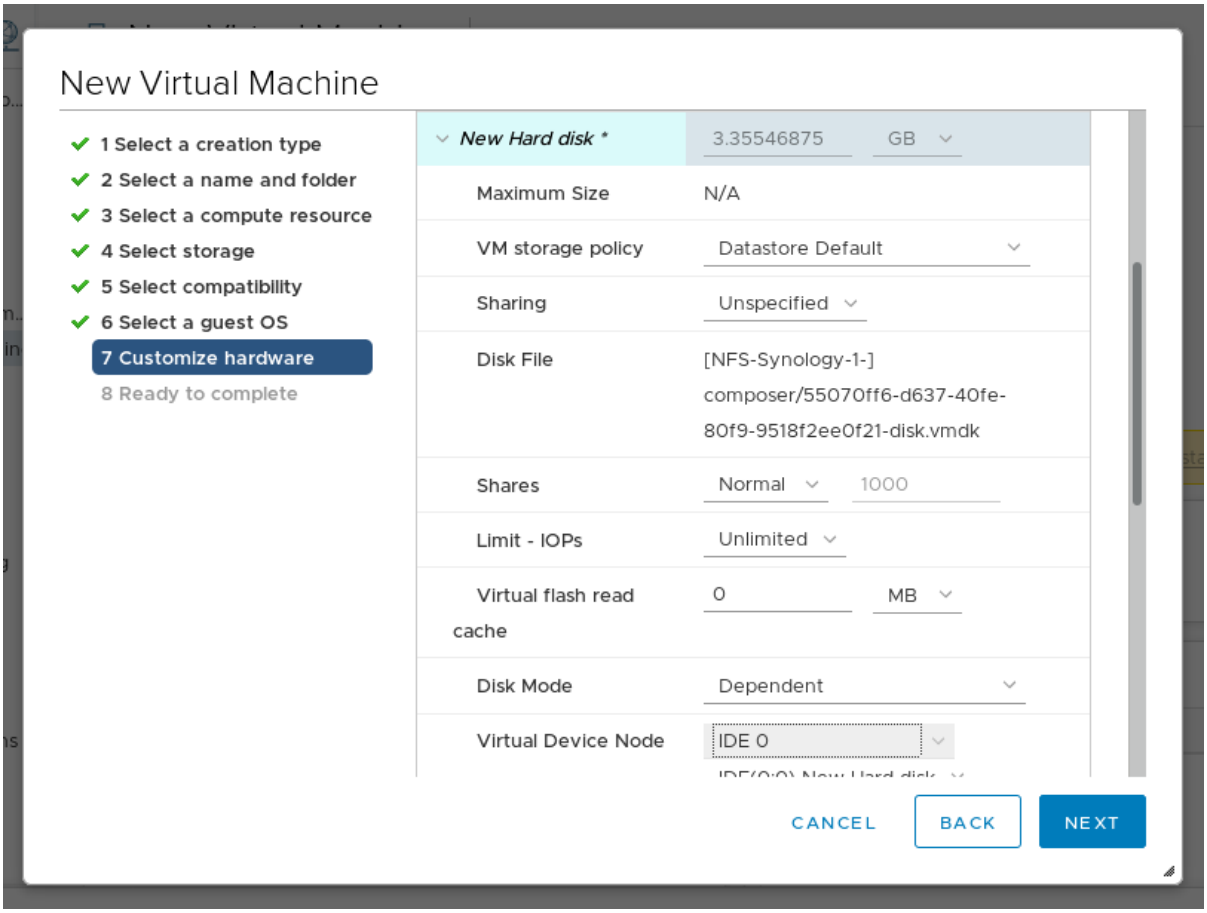
1. *HTTP 를 통해 이미지를 vSphere 에 업로드합니다. vCenter 에서 **파일 업로드** 를 클릭합니다.*




2. *VM 을 생성할 때 **장치** 구성에서 기본 **New Hard Disk** 를 삭제하고 드롭다운을 사용하여 **기존 하드 디스크 이미지** 를 선택합니다.*




3. ***IDE*** *장치를 생성한 디스크에 대한* ***가상 장치*** *노드로 사용해야 합니다. 기본 값* ***SCSI*** *로 인해 부팅 불가능한 가상 시스템이 생성됩니다.*




## *8.7. VSPHERE로 VMWARE 이미지 푸시*

VMware 이미지를 빌드하고 vSphere 인스턴스에 직접 푸시하여 이미지 파일을 다운로드하여 수동으로 푸시할 수 있습니다. 여기서는 Image Builder를 사용하여 생성한 **.vmdk** 이미지를 vSphere 인스턴스 서비스 공급자로 직접 푸시하는 단계를 설명합니다.

**사전 요구 사항**

- 시스템에 **root** 또는 **wheel** 그룹 사용자가 액세스할 수 있습니다.
- 브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열 수 있습니다.
- vSphere 계정이 있어야 합니다.

**절차**

1. **Createceilometer를 클릭합니다**.
   웹 콘솔 인터페이스에서 이미지 빌더 작업 생성을 참조하십시오.
2. 생성 중인 이미지의 일부로 원하는 구성 요소 및 패키지를 선택합니다.
3. **Commit** (커밋)을 클릭하여 프로비전에 대한 변경 사항을 커밋합니다.
   오른쪽 상단에 있는 팝업에서 저장 진행 상황을 확인한 다음 커밋한 변경 사항의 결과를 표시합니다.
4. 왼쪽 배너에서 bin **name** 링크를 클릭합니다.
5. **Customizations** (사용자 지정) 탭을 선택하여1)에 대한 사용자 계정을 만듭니다.
   청사진은 사용자 계정 생성을 참조하십시오.
6. **이미지** 탭을 선택합니다.
7. **Create Image** (이미지 만들기)를 클릭하여 사용자 지정 이미지를 만듭니다.
   Image type(이미지 유형) 창이 열립니다.
8. **이미지 유형** 창에서 다음을 수행합니다.
   a. 드롭다운 메뉴에서 유형을 선택합니다. VMware VSphere(.vmdk).
   b. 이미지를 vSphere에 업로드하려면 **Upload to VMware** 확인란을 선택합니다.
   c. 선택 사항: 인스턴스화할 이미지의 크기를 설정합니다. 최소 기본 크기는 2GB입니다.
   d. **다음**을 클릭합니다.
9. **VMware에 업로드** 창의 **Authentication** 에서 다음 세부 정보를 입력합니다.
   a. 사용자 이름: vSphere 계정의 사용자 이름입니다.
   b. 암호: vSphere 계정을 나열합니다.
10. **VMware에 업로드** 창의 **Destination** 에서 다음 세부 정보를 입력합니다.
    a. **이미지 이름**: 업로드할 이미지의 이름입니다.
    b. **호스트**: 이미지가 업로드될 VMware vSphere의 URL 입니다.
    c. **클러스터**: 이미지를 업로드할 클러스터의 이름입니다.

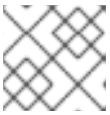
d. ***데이터 센터***: *이미지를 업로드할 데이터 센터의 이름입니다.*

e. ***데이터 저장소***: *이미지를 업로드할 데이터 저장소의 이름입니다.*

f. ***다음****을 클릭합니다.*

11. ***검토*** *창에서 이미지 생성에 대한 세부 정보를 검토하고* ***완료*** *를 클릭합니다.*
***Back*** *을 클릭하여 잘못된 세부 정보를 수정할 수 있습니다.*

*Image Builder는 RHEL vSphere 이미지의 구성 요소를 큐에 추가하고 지정한 vSphere 인스턴스의 클러스터에 이미지를 생성 및 업로드합니다.*



***참고***

*이미지 빌드 및 업로드 프로세스를 완료하는 데 몇 분이 걸립니다.*

*프로세스가 완료되면* ***이미지 빌드 완료*** *상태를 확인할 수 있습니다.*

*검증*

*이미지 상태 업로드가 성공적으로 완료되면 업로드한 이미지에서 VM(가상 머신)을 생성하여 로그인할 수 있습니다. 다음 단계를 수행합니다.*

1. *VMware vSphere Client에 액세스합니다.*
2. *지정한 vSphere 인스턴스에서 클러스터에서 이미지를 검색합니다.*
3. *업로드한 이미지에서 새 가상 머신을 생성할 수 있습니다. 이를 위해 다음을 수행합니다.*
   a. *업로드한 이미지를 선택합니다.*
   b. *선택한 이미지에서 오른쪽 버튼을 클릭합니다.*
   c. *New Virtual Machine을 클릭합니다.*
   ***New Virtual Machine*** *창이 열립니다.*

   ***New Virtual Machine*** *창에서 다음 세부 정보를 제공합니다.*

      i. *생성 유형을 선택합니다. 새 가상 머신 생성을 선택할 수 있습니다.*
      ii. *이름 및 폴더를 선택합니다. 예를 들어 가상 머신 이름: vSphere Virtual Machine 및 vSphere Client 내에서 선택한 위치입니다.*
      iii. *컴퓨터 리소스를 선택합니다. 이 작업에 대한 대상 컴퓨터 리소스를 선택합니다.*
      iv. *스토리지 선택: 예를 들어, NFS-Node1을 선택합니다.*
      v. *호환성 선택: 이미지는 BIOS 여야 합니다.*
      vi. *게스트 OS를 선택합니다. 예를 들어 Linux 및 _Red Hat Fedora(64비트)를 선택합니다.*
      vii. ***하드웨어 사용자 정의***: *VM을 생성할 때 오른쪽 상단에 있는* ***장치 구성*** *버튼의 기본 New Hard Disk를 삭제하고 드롭다운을 사용하여 기존 하드 디스크 이미지를 선택합니다.*
      viii. *완료할 준비가 된 경우: 세부 사항을 검토하고* ***완료*** *를 클릭하여 이미지를 만듭니다.*

   d. ***VM*** *탭으로 이동합니다.*

i. 목록에서 생성한 VM을 선택합니다.

ii. 패널에서 **Start** 버튼을 클릭합니다. VM 이미지 로드를 보여주는 새 창이 표시됩니다.

iii. credential에 대해 생성한 자격 증명을 사용하여 로그인합니다.

iv. credential에 추가한 패키지가 설치되어 있는지 확인할 수 있습니다. 예를 들면 다음과 같습니다.

```
$ rpm -qa | grep firefox
```

**추가 리소스**

- vSphere Client 설치.

## 8.8. AZURE 클라우드로 VHD 이미지 푸시

이미지 빌더를 사용하여 **.vhd** 이미지를 생성할 수 있습니다. 그런 다음 **.vhd** 이미지를 Azure Cloud 서비스 공급자의 Blob Storage로 푸시할 수 있습니다.

**사전 요구 사항**

- 시스템에 대한 루트 액세스 권한이 있어야 합니다.
- 브라우저에서 RHEL 웹 콘솔의 Image Builder 인터페이스를 열었습니다.
- 스토리지 계정이 생성되어 있어야 합니다.
- 쓰기 가능한 Blob 스토리지 가 있어야 합니다.

**절차**

1. **Createceilometer**를 클릭하여 프로비전을 생성합니다. 웹 콘솔 인터페이스에서 이미지 빌더 작업 생성을 참조하십시오.
2. 생성 중인 이미지의 일부로 원하는 구성 요소 및 패키지를 선택합니다.
3. **Commit** (커밋)을 클릭하여 프로비전에 대한 변경 사항을 커밋합니다.
   오른쪽 상단에 있는 작은 팝업에서 저장 진행 상황을 확인한 다음 커밋한 변경 사항의 결과를 표시합니다.
4. 왼쪽 배너에서 bin **name** 링크를 클릭합니다.
5. **Images** (이미지) 탭을 선택합니다.
6. **Create Image** (이미지 만들기)를 클릭하여 사용자 지정 이미지를 만듭니다.
   팝업 창이 열립니다.
   a. **"Type"** 드롭다운 메뉴 목록에서 **Azure Disk Image (.vhd)** 이미지를 선택합니다.
   b. **"Azure로 업로드"** 확인란을 선택하여 이미지를 Azure 클라우드에 업로드하고 **다음을** 클릭합니다.
   c. Azure에 대한 액세스를 인증하려면 해당 필드에 "Storage account" 및 "Storage access key"를 입력합니다. **다음**을 클릭합니다.

Storage 계정 세부 정보는 SettingsReplicasAccess Key 메뉴 목록에서 확인할 수 있습니다.

d. 업로드할 이미지 파일과 이미지를 내보낼 이미지 파일에 사용할 **"이미지 이름"** 을 입력합니다. **다음**을 클릭합니다.

e. 제공한 정보를 검토하고 **완료** 를 클릭합니다.
경우에 따라 **뒤로** 클릭하여 잘못된 세부 정보를 수정할 수 있습니다.

7. 오른쪽 상단에는 이미지 생성 프로세스가 메시지로 시작될 때 표시되는 작은 팝업이 표시됩니다. "이미지 생성이 큐에 추가되었습니다.
이미지 프로세스 생성이 완료되면 이미지를 생성한 images를 클릭합니다. 이미지 탭에서 생성한 이미지의 **"이미지 빌드 완료"** 상태를 확인할 수 있습니다.

8. **Azure Cloud** 로 내보낸 이미지에 액세스하려면 **Azure** 포털에 액세스합니다.

9. 검색바에서 **Images** 를 입력하고 **Services** (서비스)에서 첫 번째 항목을 선택합니다. 이미지 대시보드 로 리디렉션됩니다.

10. **+**추가를 클릭합니다. **Create an Image dashboard**로 리디렉션됩니다.

아래 세부 정보를 삽입합니다.

a. **이름: 새 이미지의 이름을 선택합니다.**

b. **리소스 그룹: 리소스 그룹을 선택합니다.**

c. **위치: 스토리지 계정에 할당된 리전과 일치하는 위치를 선택합니다. 그렇지 않으면 Blob** 을 선택할 수 없습니다.

d. **OS 유형: OS 유형을 Linux** 로 설정합니다.

e. **VM 생성: Gen 1** 에 **VM** 생성을 설정합니다.

f. 스토리지 **Blob: Storage blob** 입력 오른쪽에 있는 찾아보기 를 클릭합니다. 대화 상자를 사용하여 이전에 업로드한 이미지를 찾습니다.

나머지 필드는 기본 선택에 그대로 유지합니다.

11. 생성 을 클릭하여 이미지를 생성합니다. 이미지가 생성되면 오른쪽 상단에 **"Successfully created image"** 라는 메시지를 볼 수 있습니다.

12. 새로 고침 을 클릭하여 새 이미지를 확인하고 새로 생성된 이미지를 엽니다.

13. **+ Create VM** 을 클릭합니다. 가상 머신 대시보드 생성으로 리디렉션됩니다.

14. 기본 탭의 프로젝트 세부 정보에서 ***Subscription** 및 **Resource Group** 은 이미 사전 설정됩니다.

새 리소스 그룹을 생성하려는 경우

a. **Create new** 를 클릭합니다.

**Resource Group Name** 컨테이너를 생성하라는 팝업 메시지가 표시됩니다.

b. 이름을 입력하고 확인 을 클릭합니다.

이미 설정된 리소스 그룹을 유지하려면 다음을 수행하십시오.

15. 인스턴스 세부 정보 에서 다음을 삽입합니다.

a. 가상 머신 이름

b. 리전

c. 이미지: 생성한 이미지는 기본적으로 사전 선택됩니다.

d.

*크기: 요구 사항에 더 적합한 VM 크기를 선택합니다.*

*나머지 필드는 기본 선택에 그대로 유지합니다.*

16.

*관리자 계정에서 아래 세부 정보를 입력합니다.*

a.

*사용자 이름: 계정 관리자의 이름입니다.*

b.

*SSH 공개 키 소스: 드롭다운 메뉴에서 새 키 쌍 생성 을 선택합니다.*

*이미 보유하고 있는 키 쌍을 사용하거나 새 키 쌍을 만들 수 있습니다. 또는 이미지 빌더를 사용하여 사전 정의된 공개 키가 있는 이미지에 사용자를 추가할 수 있습니다. 자세한 내용은 SSH 키를 사용하여 사용자 계정 생성 을 참조하십시오.*

c.

*키 쌍 이름: 키 쌍의 이름을 삽입합니다.*

17.

*인바운드 포트 규칙에서 다음을 선택합니다.*

a.

*공용 인바운드 포트: 선택한 포트를 허용합니다.*

b.

*인바운드 포트 선택: 기본 세트 SSH(22 )를 사용합니다.*

18.

*검토 + 생성을 클릭합니다. 검토 + 생성 탭으로 리디렉션되어 검증이 통과했음을 확인합니다.*

19.

*세부 사항을 검토하고 생성 을 클릭합니다.*

*선택적으로 이전 옵션을 클릭하여 선택한 이전 옵션을 수정할 수 있습니다.*

20.

*팝업에서 새 키 쌍 창을 생성합니다. 개인 키 다운로드를 클릭하고 리소스를 생성합니다.*

*키 파일을 "Key.pem"으로 저장합니다.*

21. 배포가 완료된 후 리소스로 이동을 클릭합니다.

22. VM 세부 정보를 사용하여 새 창이 리디렉션됩니다. 페이지 오른쪽 상단에 있는 공용 IP 주소를 선택하여 클립보드에 복사합니다.

이제 가상 머신에 연결할 VM과 SSH 연결을 생성하려면 다음을 수행합니다.

1. 터미널을 엽니다.

2. 프롬프트에서 가상 머신에 대한 SSH 연결을 엽니다. IP 주소를 VM에서 가져온 주소로 교체하고 .pem의 경로를 키 파일이 다운로드한 경로로 바꿉니다.

```
# ssh -i ./Downloads/yourKey.pem azureuser@10.111.12.123
```

3. 계속 연결하려는지 확인해야 합니다. 계속하려면 yes를 입력합니다.

결과적으로 Azure Storage Blob으로 내보낸 출력 이미지를 프로비저닝할 준비가 되었습니다.

추가 리소스

- Azure Storage 설명서.
- Azure Storage 계정을 생성합니다.
- Red Hat 고객 포털에서 케이스를 엽니다.
- 도움말 + 지원.
- Red Hat에 문의하기.

### 8.9. OPENSTACK에 QCOW2 이미지 업로드

이미지 빌더는 OpenStack 클라우드 배포에 업로드하는 데 적합한 이미지를 생성하고 해당 환경에서 인스턴스를 시작할 수 있습니다. 이에서는 QCOW2 이미지를 OpenStack에 업로드하는 단계를 설명합니다.

사전 요구 사항

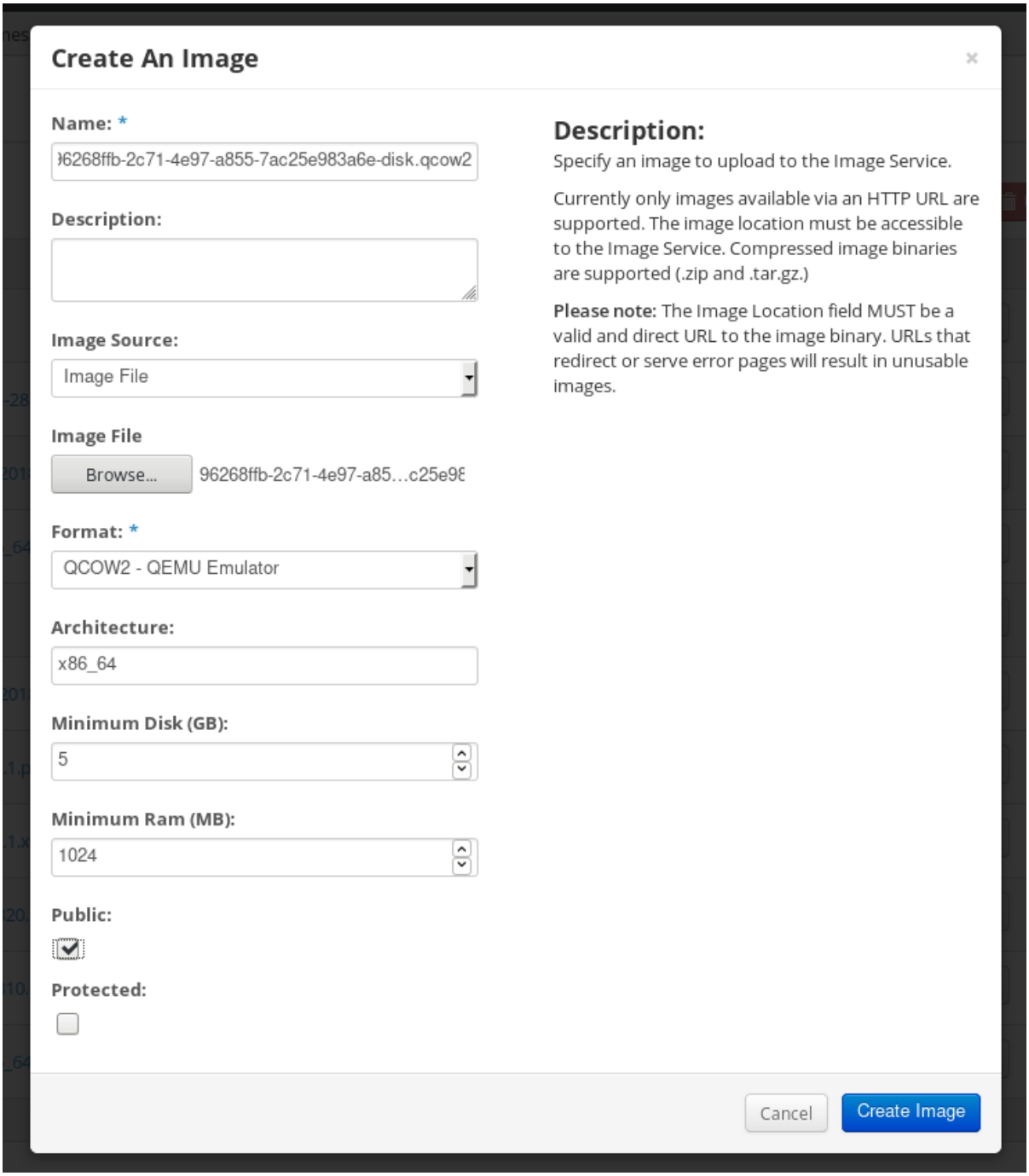
- 이미지 빌더에서 생성한 OpenStack별 이미지가 있어야 합니다. 이미지를 생성할 때 GUI에서 CLI 또는 OpenStack Image(.qcow2) 에서 openstack output type을 사용합니다.

주의

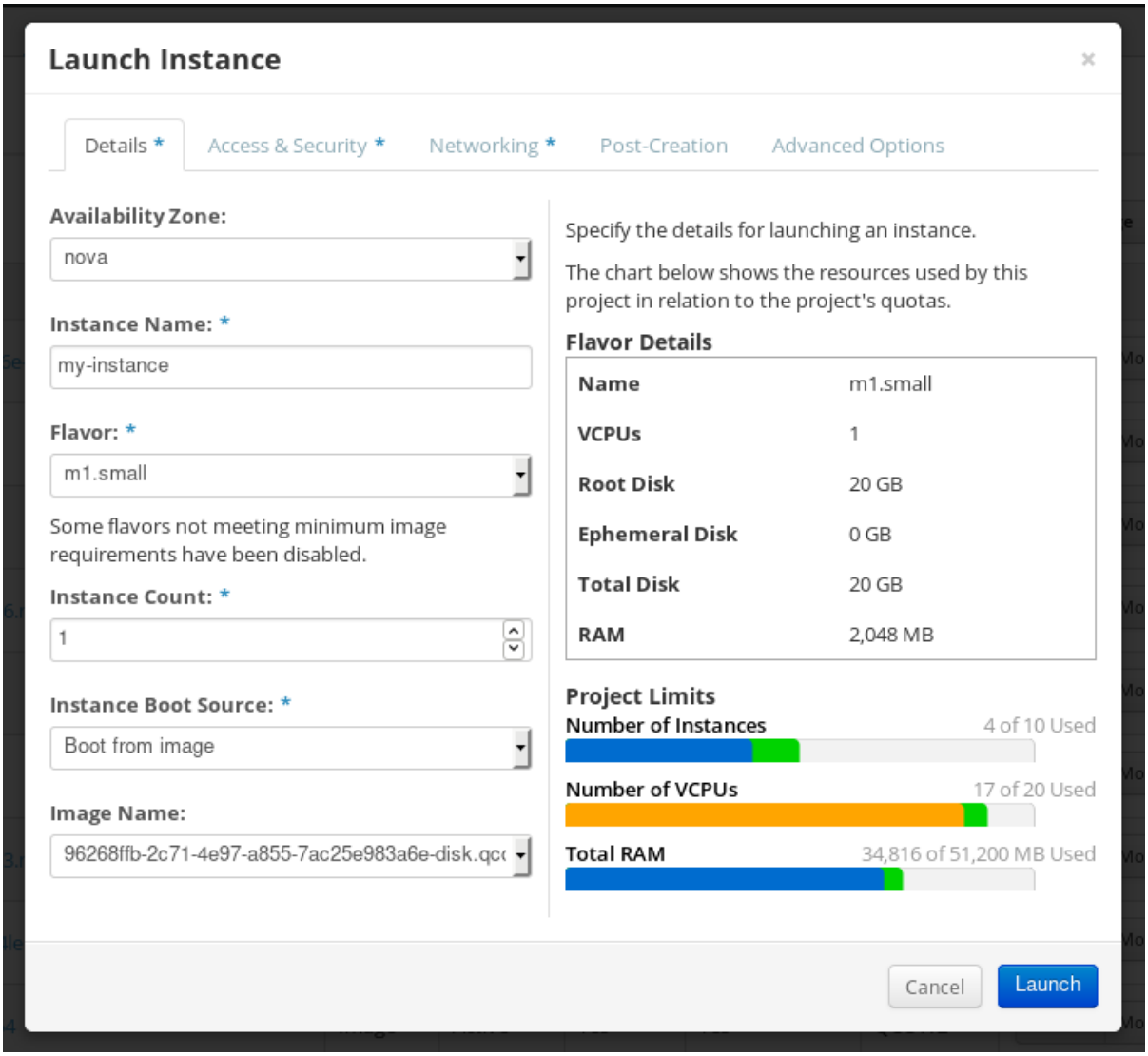
이미지 빌더는 qcow2 또는 QEMU QCOW2 이미지(.qcow2)로 일반 QCOW2 이미지 유형 출력 형식도 제공합니다. QCOW2 형식이기도 하지만 OpenStack과 관련된 추가 변경 사항이 포함된 OpenStack 이미지 유형으로 실수하지 마십시오.

절차

1. 이미지를 OpenStack에 업로드하고 해당 이미지에서 인스턴스를 시작합니다. 이미지 인터페이스를 사용하여 다음을 수행합니다.




2. *해당 이미지로 인스턴스를 시작합니다.*




3. *스냅샷에서 모든 메커니즘(CLI 또는 **OpenStack Web UI**)을 사용하여 인스턴스를 실행할 수 있습니다. **SSH**를 통해 개인 키를 사용하여 결과 인스턴스에 액세스합니다. **cloud-user** 로 로그인합니다.*

## *8.10. RUNTIMECLASS에 이미지 업로드 준비*

*이 섹션에서는 **vGPU Cloud**에 배포할 수 있는 사용자 지정 이미지를 확인하는 단계에 대해 설명합니다. **images will need a specific configuration to boot successfully, because you need the custom images to meet certain requirements before you use it.**라는 메시지가 성공적으로 부팅되기 때문에 **images will need a specific configuration to boot successfully, because you request the custom images to meet certain requirements before you use it.** 이를 위해 **RuntimeClass image_check** 도구를 사용하는 것이 좋습니다.*

참고

사용자 정의 이미지 확인은 선택적 작업입니다. 이미지 빌더는 **RuntimeClass**의 요구 사항을 준수하는 이미지를 생성합니다.

사전 요구 사항

- 이미지 빌더에서 생성한 **RuntimeClass** 이미지가 있어야 합니다.

절차

1. 확인할 이미지가 포함된 시스템에 연결합니다. **image_check** 툴.

2. **image_check** 툴을 다운로드합니다.

```
$ curl -O http://docs-aliyun.cn-hangzhou.oss.aliyun-
inc.com/assets/attach/73848/cn_zh/1557459863884/image_check
```

3. 이미지 규정 준수 툴의 파일 권한을 변경합니다.

```
# chmod +x image_check
```

4. 명령을 실행하여 이미지 규정 준수 툴 검사를 시작합니다.

```
# ./image_check
```

툴은 시스템 구성을 확인하고 화면에 표시되는 보고서를 생성합니다. **image_check** 툴은 이 보고서를 이미지 규정 준수 도구가 실행 중인 동일한 폴더에 저장합니다.

5. 탐지 항목 중 하나라도 실패하면 지침에 따라 수정합니다. 자세한 내용은 링크를 참조하십시오. 탐지 항목 섹션.

추가 리소스

- 이미지 규정 준수 도구

### 8.11. RUNTIMECLASS에 이미지 업로드

이 섹션에서는 개체 스토리지 서비스(OSS)에 RuntimeClass 이미지를 업로드하는 방법에 대해 설명합니다.

사전 요구 사항

- 시스템이 rhcos 이미지를 업로드하도록 설정되어 있습니다.
- 이미지 빌더에서 생성한 RuntimeClass 이미지가 있어야 합니다. 이미지를 생성할 때 RHEL 7 또는 RHEL 8의 ami 출력 유형을 사용합니다.
- 버킷이 있습니다. 버킷 생성을 참조하십시오.
- 활성 vGPU 계정이 있습니다.
- OSS 를 활성화했습니다.

절차

1. OSS 콘솔에 로그인합니다.
2. 왼쪽 Bucket 메뉴에서 이미지를 업로드할 버킷을 선택합니다.
3. 오른쪽 상단 메뉴에서 파일 탭을 클릭합니다.
4. 업로드를 클릭합니다. 오른쪽 창에서 창 대화 상자가 열립니다. 다음 정보를 선택합니다.
   - 업로드 대상: 파일을 현재 디렉터리 또는 지정된 디렉터리에 업로드하도록 선택합니다.
   - 파일 ACL: 업로드된 파일의 권한 유형을 선택합니다.

5. *업로드를 클릭합니다.*

6. *업로드할 이미지를 선택합니다.*

7. *열기를 클릭합니다.*

*결과적으로 사용자 지정 이미지가 OSS Console에 업로드됩니다.*

*추가 리소스*

- *오브젝트 업로드*
- *사용자 정의 이미지에서 인스턴스 생성*
- *이미지 가져오기*

### *8.12. RUNTIMECLASS로 이미지 가져오기*

*이 섹션에서는 RuntimeClass 이미지를 Elastic Cloud Console(ECS)으로 가져오는 방법에 대해 설명합니다.*

*사전 요구 사항*

- *이미지를 Object Storage Service(OSS)에 업로드했습니다.*

*절차*

1. *SriovIBNetwork 콘솔에 로그인합니다.*

   i. *왼쪽 메뉴에서 이미지를 클릭합니다.*

ii. *오른쪽 상단에서 이미지 가져오기 를 클릭합니다. 창 대화 상자가 열립니다.*

iii. *이미지가 있는 올바른 리전을 설정했는지 확인합니다. 다음 정보를 입력합니다.*

a. ***OSS** 오브젝트 주소: **OSS** 개체 주소를 얻는 방법을 참조하십시오.*

b. *이미지 이름:*

c. *운영 체제:*

d. *시스템 디스크 크기:*

e. *시스템 아키텍처:*

f. *플랫폼: **Red Hat***

iv. *선택적으로 다음 세부 정보를 제공합니다.*

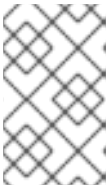
g. *이미지 형식: 업로드된 이미지 형식에 따라 **qcow2** 또는 **ami**입니다.*

h. *이미지 설명:*

i. *데이터 디스크 이미지 추가:*

*왼쪽 메뉴에서 필요한 버킷을 선택한 후 **OSS** 관리 콘솔에서 주소를 결정하고 파일 섹션을 선택한 다음 적절한 이미지에 대한 오른쪽에 있는 세부 정보 링크를 클릭합니다. 화면 오른쪽에 창이 표시되고 이미지 세부 정보가 표시됩니다. **OSS** 개체 주소는 **URL** 상자에 있습니다.*

2. ***OK**를 클릭합니다.*



참고

가져오기 프로세스 시간은 이미지 크기에 따라 달라질 수 있습니다.

결과적으로 사용자 지정 이미지를 **SriovIBNetwork Console**로 가져옵니다. 사용자 지정 이미지에서 인스턴스를 생성할 수 있습니다.

추가 리소스

- 이미지 가져오기를 위한 참고 사항
- 사용자 정의 이미지에서 인스턴스 생성
- 오브젝트 업로드

### 8.13. RUNTIMECLASS를 사용하여 사용자 지정 이미지의 인스턴스 생성

사용자 지정 이미지의 인스턴스를 생성할 수 있습니다.

사전 요구 사항

- **OSS** 를 활성화하여 사용자 지정 이미지를 업로드했습니다.
- 이미지를**octets Console**로 가져왔습니다.

절차

1. **SriovIBNetwork** 콘솔에 로그인합니다.
2. 왼쪽 메뉴에서 **Instances** 를 선택합니다.
3. 상단 모서리에서 인스턴스 생성 을 클릭합니다. 새 창으로 리디렉션됩니다.

4. *필요한 모든 정보를 입력합니다. 자세한 내용은 마법사를 사용하여 인스턴스 생성 을 참조하십시오.*

5. *인스턴스 생성 을 클릭하고 순서를 확인합니다.*

*참고*

*서브스크립션에 따라 인스턴스 생성 대신 주문 생성 옵션을 확인할 수 있습니다.*

*따라서 배포를 위한 활성 인스턴스를 사용할 수 있습니다.*

*추가 리소스*

- *사용자 지정 이미지를 사용하여 인스턴스 생성*
- *마법사를 사용하여 인스턴스 생성*