# Red Hat Enterprise Linux 8

## 사용자 지정 RHEL 시스템 이미지 구성

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

Last Updated: 2024-07-20

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

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

## 법적 공지



## 초록

RHEL 이미지 빌더는 설치 디스크, 가상 머신, 클라우드 벤더별 이미지 등 배포 준비 사용자 지정 시스템 이미지를 생성하는 툴입니다. RHEL 이미지 빌더를 사용하면 각 출력 유형에 필요한 특정 구성이 제거되므로 수동 프로시저와 비교하여 이러한 이미지를 더 빠르게 생성할 수 있습니다.

# 차례

# RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

**Jira를 통해 피드백 제출 (등록 필요)**

1. Jira 웹 사이트에 로그인합니다.
2. 상단 탐색 모음에서 **생성** 을 클릭합니다.
3. **요약** 필드에 설명 제목을 입력합니다.
4. **설명** 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. 대화 상자 하단에서 **생성** 을 클릭합니다.

# 1장. RHEL 이미지 빌더 설명

시스템을 배포하려면 시스템 이미지를 생성합니다. RHEL 시스템 이미지를 생성하려면 RHEL 이미지 빌더 툴을 사용합니다. RHEL 이미지 빌더를 사용하여 클라우드 플랫폼에 배포할 수 있는 시스템 이미지를 포함하여 RHEL의 사용자 지정 시스템 이미지를 생성할 수 있습니다. RHEL 이미지 빌더는 각 출력 유형에 대한 설정 세부 정보를 자동으로 처리하므로 이미지 생성 방법보다 사용하기 쉽고 빠르게 작업할 수 있습니다. **composer-cli** 툴의 명령줄 인터페이스 또는 RHEL 웹 콘솔에서 그래픽 사용자 인터페이스를 사용하여 RHEL 이미지 빌더 기능에 액세스할 수 있습니다.

**참고**

RHEL 8.3 이후 **osbuild-composer** 백엔드는 **lorax-composer** 를 대체합니다. 새 서비스는 이미지 빌드를 위한 REST API를 제공합니다.

## 1.1. RHEL 이미지 빌더 용어

RHEL 이미지 빌더에서는 다음 개념을 사용합니다.

**Quarkus**

사용자 지정 시스템 이미지에 대한 설명입니다. 시스템의 일부가 될 패키지 및 사용자 정의가 나열됩니다. 사용자 지정으로 done을 편집하고 특정 버전으로 저장할 수 있습니다. 블루프린트에서 시스템 이미지를 생성하면 이미지가 RHEL 이미지 빌더 인터페이스의 블루프린트와 연결됩니다.
TOML 형식으로 블루프린트를 생성합니다.

**compose**

작문은 특정 버전의 특정 버전에 따라 시스템 이미지의 개별 빌드입니다. 용어로 작성은 시스템 이미지, 생성, 입력, 메타데이터 및 프로세스 자체의 로그를 나타냅니다.

**사용자 정의**

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

## 1.2. RHEL 이미지 빌더 출력 형식

RHEL 이미지 빌더는 다음 표에 표시된 여러 출력 형식으로 이미지를 생성할 수 있습니다.

**표 1.1. RHEL 이미지 빌더 출력 형식**

| 설명 | CLI 이름 | 파일 확장자 |
|---|---|---|
| QEMU 이미지 | **qcow2** | **.qcow2** |
| 디스크 아카이브 | **tar** | **.tar** |
| Amazon Web Services | **raw** | **.raw** |
| Microsoft Azure | **vhd** | **.vhd** |
| Google Cloud Platform | **gce** | **.tar.gz** |
| VMware vSphere | **vmdk** | **.vmdk** |

| 설명 | CLI 이름 | 파일 확장자 |
|---|---|---|
| VMware vSphere | **OVA** | **.ova** |
| 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 Image | **edge-raw-image** | **.raw.xz** |
| Edge Simplified Installer용 RHEL | **edge-simplified-installer** | **.iso** |
| RHEL for Edge AMI | **edge-ami** | **.ami** |
| RHEL for Edge VMDK | **edge-vsphere** | **.vmdk** |
| RHEL 설치 프로그램 | **image-installer** | **.iso** |
| Oracle Cloud Infrastructure | **.oci** | **.qcow2** |

지원되는 유형을 확인하려면 명령을 실행합니다.

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

# 2장. RHEL 이미지 빌더 설치

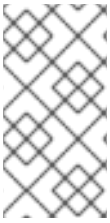
RHEL 이미지 빌더를 사용하기 전에 설치해야 합니다.

## 2.1. RHEL 이미지 빌더 시스템 요구 사항

RHEL 이미지 빌더를 실행하는 호스트는 다음 요구 사항을 충족해야 합니다.

**표 2.1. RHEL 이미지 빌더 시스템 요구 사항**

| 매개변수 | 최소 필수 값 |
|---|---|
| 시스템 유형 | 전용 호스트 또는 가상 머신. Red Hat UBI(Universal Base Images)를 포함한 컨테이너에서 RHEL 이미지 빌더는 지원되지 않습니다. |
| 프로세서 | 2개의 코어 |
| 메모리 | 4GiB |
| 디스크 공간 | '/var/cache/' 파일 시스템에서 20GiB의 여유 공간 |
| 액세스 권한 | root |
| 네트워크 | Red Hat CDN(Content Delivery Network)에 대한 인터넷 연결. |



**참고**

인터넷 연결이 없는 경우 격리된 네트워크에서 RHEL 이미지 빌더를 사용합니다. 이를 위해 Red Hat CDN(Content Delivery Network)에 연결되지 않도록 로컬 리포지토리를 가리키도록 기본 리포지토리를 재정의해야 합니다. 콘텐츠가 내부적으로 미러링되었는지 확인하거나 Red Hat Satellite를 사용하십시오.

**추가 리소스**

- RHEL 이미지 빌더 리포지토리 구성
- Red Hat 이미지 빌더 이미지를 사용하여 Satellite에 프로비저닝

## 2.2. RHEL 이미지 빌더 설치

모든 **osbuild-composer** 패키지 기능에 액세스할 수 있도록 RHEL 이미지 빌더를 설치합니다.

**사전 요구 사항**

- RHEL 이미지 빌더를 설치하려는 RHEL 호스트에 로그인되어 있습니다.
- 호스트는 RHSM(Red Hat Subscription Manager) 또는 Red Hat Satellite에 가입되어 있습니다.
- **BaseOS** 및 **AppStream** 리포지토리를 활성화하여 RHEL 이미지 빌더 패키지를 설치할 수 있습니다.

**절차**

1. RHEL 이미지 빌더 및 기타 필요한 패키지를 설치합니다.

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

   - **osbuild-composer** - 사용자 지정된 RHEL 운영 체제 이미지를 빌드하는 서비스입니다.
   - **composer-cli**- 이 패키지를 사용하면 CLI 인터페이스에 액세스할 수 있습니다.
   - **Cockpit-composer** - 이 패키지를 사용하면 웹 UI 인터페이스에 액세스할 수 있습니다. 웹 콘솔은 **cockpit-composer** 패키지의 종속성으로 설치됩니다.

2. RHEL 이미지 빌더 소켓을 활성화하고 시작합니다.

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

3. 웹 콘솔에서 RHEL 이미지 빌더를 사용하려면 활성화한 후 시작합니다.

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

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

4. 로그아웃하지 않고 **composer-cli** 명령의 자동 완성 기능이 즉시 작동하도록 쉘 구성 스크립트를 로드합니다.

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

5. RHEL 호스트에서 실행 중인 **osbuild-composer** 서비스를 다시 시작합니다.

```
# systemctl restart osbuild-composer
```

**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**

- **composer-cli**:을 실행하여 설치가 작동하는지 확인합니다.

```
# composer-cli status show
```

**문제 해결**

시스템 저널을 사용하여 RHEL 이미지 빌더 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
```

## 2.3. LORAX-COMPOSER RHEL 이미지 빌더 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
# yum remove weldr-client
```

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 weldr-client
```

3. **lorax-composer** 패키지를 설치합니다.

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

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. RHEL 이미지 빌더 리포지토리 구성

RHEL 이미지 빌더를 사용하려면 리포지토리가 구성되어 있는지 확인해야 합니다. RHEL 이미지 빌더에서 다음 유형의 리포지토리를 사용할 수 있습니다.

**공식 리포지토리 덮어쓰기**

Red Hat CDN(Content Delivery Network) 공식 리포지토리(예: 네트워크의 사용자 지정 미러) 이외의 위치에서 기본 시스템 RPM을 다운로드하려는 경우 이를 사용합니다. 공식 리포지토리 덮어쓰기를 사용하면 기본 리포지토리가 비활성화되고 사용자 지정 미러에 필요한 모든 패키지가 포함되어야 합니다.

**사용자 정의 타사 리포지토리**

이를 사용하여 공식 RHEL 리포지토리에서 사용할 수 없는 패키지를 포함합니다.

## 3.1. RHEL 이미지 빌더에 사용자 지정 타사 리포지토리 추가

사용자 지정 타사 소스를 리포지토리에 추가하고 **composer-cli** 를 사용하여 이러한 리포지토리를 관리할 수 있습니다.

**사전 요구 사항**

- 사용자 지정 타사 리포지토리의 URL이 있습니다.

**절차**

1. **/root/repo.toml** 와 같은 리포지토리 소스 파일을 만듭니다. 예를 들면 다음과 같습니다.

```
id = "k8s"
name = "Kubernetes"
type = "yum-baseurl"
url = "https://server.example.com/repos/company_internal_packages/"
check_gpg = false
check_ssl = false
system = false
```

**type** 필드에는 **yum-baseurl**,**yum-mirrorlist**, **yum-metalink** 라는 유효한 값을 사용할 수 있습니다.

2. 파일을 TOML 형식으로 저장합니다.
3. RHEL 이미지 빌더에 새 타사 소스를 추가합니다.

```
$ composer-cli sources add <file-name>.toml
```

**검증**

1. 새 소스가 성공적으로 추가되었는지 확인합니다.

```
$ composer-cli sources list
```

2. 새 소스 콘텐츠를 확인합니다.

```
$ composer-cli sources info <source_id>
```

## 3.2. RHEL 이미지 빌더에 특정 배포판을 사용하여 타사 리포지토리 추가

선택 사항 필드 **distro** 를 사용하여 사용자 지정 타사 소스 파일에서 배포 목록을 지정할 수 있습니다. 리포지토리 파일은 이미지 빌드 중에 종속성을 확인하는 동안 배포 문자열 목록을 사용합니다.

**rhel-8** 을 지정하는 모든 요청에서는 이 소스를 사용합니다. 예를 들어 패키지를 나열하고 **rhel-8** 을 지정하는 경우 이 소스가 포함됩니다. 그러나 호스트 배포에 대한 패키지를 나열해도 이 소스는 포함되지 않습니다.

**사전 요구 사항**

- 사용자 지정 타사 리포지토리의 URL이 있습니다.
- 지정할 배포 목록이 있습니다.

**절차**

1. **/root/repo.toml** 와 같은 리포지토리 소스 파일을 만듭니다. 예를 들어 배포를 지정하려면 다음을 수행합니다.

```
check_gpg = true
check_ssl = true
distros = ["rhel-8"]
id = "rh9-local"
name = "packages for RHEL"
system = false
type = "yum-baseurl"
url = "https://local/repos/rhel8/projectrepo/"
```

2. 파일을 TOML 형식으로 저장합니다.
3. RHEL 이미지 빌더에 새 타사 소스를 추가합니다.

```
$ composer-cli sources add <file-name>.toml
```

**검증**

1. 새 소스가 성공적으로 추가되었는지 확인합니다.

```
$ composer-cli sources list
```

2. 새 소스 콘텐츠를 확인합니다.

```
$ composer-cli sources info <source_id>
```

## 3.3. GPG를 사용하여 리포지토리 메타데이터 확인

손상된 패키지를 감지하고 방지하려면 DNF 패키지 관리자를 사용하여 RPM 패키지에서 GNU Privacy Guard(GPG) 서명을 확인하고, 리포지토리 메타데이터가 GPG 키로 서명되었는지 확인할 수 있습니다.

키 URL로 **gpgkeys** 필드를 설정하여 **https** 를 통해 확인할 **gpgkey** 를 입력할 수 있습니다. 또는 보안을 개선하기 위해 전체 키를 **gpgkeys** 필드에 삽입하여 URL에서 키를 가져오는 대신 직접 가져올 수도 있습니다.

**사전 요구 사항**

- 리포지토리로 사용할 디렉터리가 존재하고 패키지가 포함되어 있습니다.

**절차**

1. 리포지토리를 생성할 폴더에 액세스합니다.

```
$ cd repo/
```

2. **createrepo_c** 를 실행하여 RPM 패키지에서 리포지토리를 생성합니다.

```
$ createrepo_c .
```

3. repodata가 있는 디렉터리에 액세스합니다.

```
$ cd repodata/
```

4. **repomd.xml** 파일에 서명합니다.

```
$ gpg -u <_gpg-key-email_> --yes --detach-sign --armor /srv/repo/example/repomd.xml
```

5. 리포지토리에서 GPG 서명 검사를 활성화하려면 다음을 수행합니다.

   a. 리포지토리 소스에서 **check_repogpg = true** 를 설정합니다.

   b. 검사를 수행할 **gpgkey** 를 입력합니다. **https** 를 통해 키를 사용할 수 있는 경우 키 URL을 사용하여 **gpgkeys** 필드를 설정합니다. 필요한 만큼 URL 키를 추가할 수 있습니다.
   다음은 예제입니다.

```
check_gpg = true
check_ssl = true
id = "signed local packages"
name = "repository_name"
type = "yum-baseurl"
url = "https://local/repos/projectrepo/"
check_repogpg = true
gpgkeys=["https://local/keys/repokey.pub"]
```

   또는 **gpgkeys** 필드에 직접 GPG 키를 추가합니다. 예를 들면 다음과 같습니다.

```
check_gpg = true
check_ssl = true
check_repogpg
id = "custom-local"
name = "signed local packages"
type = "yum-baseurl"
url = "https://local/repos/projectrepo/"
gpgkeys=["https://remote/keys/other-repokey.pub",
'"-----BEGIN PGP PUBLIC KEY BLOCK-----
…
-----END PGP PUBLIC KEY BLOCK-----"']
```

   - 테스트에서 서명을 찾을 수 없는 경우 GPG 툴에 다음과 유사한 오류가 표시됩니다.

```
$ GPG verification is enabled, but GPG signature is not available.
This may be an error or the repository does not support GPG verification:
Status code: 404 for http://repo-server/rhel/repodata/repomd.xml.asc (IP:
192.168.1.3)
```

- 서명이 유효하지 않은 경우 GPG 툴에 다음과 유사한 오류가 표시됩니다.

```
repomd.xml GPG signature verification error: Bad GPG signature
```

**검증**

- 리포지토리의 서명을 수동으로 테스트합니다.

```
$ gpg --verify /srv/repo/example/repomd.xml.asc
```

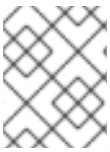
## 3.4. RHEL 이미지 빌더 공식 리포지토리 덮어쓰기

RHEL 이미지 빌더 **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. 여기에는 추가 소프트웨어를 설치하거나 이미 설치된 프로그램을 최신 버전으로 업데이트하는 기본 시스템 RPM이 포함된 Red Hat 공식 리포지토리가 포함됩니다. 공식 리포지토리를 재정의하려면 **/etc/osbuild-composer/repositories/** 에서 재정의를 정의해야 합니다. 이 디렉터리는 사용자 정의 덮어쓰기를 위한 것이며 여기에 있는 파일은 **/usr/share/osbuild-composer/repositories/** 디렉터리에 있는 파일보다 우선합니다.

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

## 3.5. 시스템 리포지토리 덮어쓰기

**/etc/osbuild-composer/repositories** 디렉터리에서 RHEL 이미지 빌더에 대한 자체 리포지토리 덮어쓰기를 구성할 수 있습니다.



**참고**

RHEL 8.5 릴리스 이전에는 리포지토리 덮어쓰기는 **rhel-8.json** 입니다. RHEL 8.5부터 이름은 마이너 버전인 **rhel-84.json**,**rhel-85.json** 등도 따릅니다.

**사전 요구 사항**

- 호스트 시스템에서 액세스할 수 있는 사용자 지정 리포지토리가 있습니다.

**절차**

1. 리포지토리 덮어쓰기를 저장할 **/etc/osbuild-composer/repositories/** 디렉터리를 만듭니다.

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

2. RHEL 버전에 해당하는 이름을 사용하여 JSON 파일을 생성합니다. 또는 **/usr/share/osbuild-composer/** 에서 배포할 파일을 복사하고 해당 콘텐츠를 수정할 수 있습니다.
RHEL 8.9의 경우 **/etc/osbuild-composer/repositories/rhel-89.json** 을 사용합니다.

3. JSON 파일에 다음 구조를 추가합니다. 문자열 형식으로 다음 속성 중 하나만 지정합니다.

   - **baseurl** - 리포지토리의 기본 URL입니다.
   - **metalink** - 유효한 미러 리포지토리 목록이 포함된 metalink 파일의 URL입니다.
   - **mirrorlist** - 유효한 미러 저장소 목록이 포함된 미러 목록 파일의 URL입니다. 나머지 필드(예: **gpgkey** ) 및 **metadata_expire** 는 선택 사항입니다.
     예를 들면 다음과 같습니다.

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

     또는 **rhel-version.json** 을 RHEL 버전으로 교체하여 배포에 대한 JSON 파일을 복사할 수 있습니다(예: rhel-8.json).

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

4. 선택 사항: JSON 파일을 확인합니다.

   ```
   $ json_verify  /etc/osbuild-composer/repositories/<file>.json
   ```

5. **rhel-8.json** 파일에서 **baseurl** 경로를 편집하여 저장합니다. 예를 들면 다음과 같습니다.

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

6. **osbuild-composer.service** 를 다시 시작합니다.

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

**검증**

- 리포지토리가 올바른 URL을 가리키는지 확인합니다.

  ```
  $ cat /etc/yum.repos.d/redhat.repo
  ```

  리포지토리가 **/etc/yum.repos.d/redhat.repo** 파일에서 복사되는 올바른 URL을 가리키는 것을 확인할 수 있습니다.

**추가 리소스**

- 리포지토리에서 사용할 수 있는 최신 RPM 버전은 **osbuild-composer** 에 표시되지 않습니다.

## 3.6. 서브스크립션이 필요한 시스템 리포지토리 덮어쓰기

**/etc/yum.repos.d/redhat.repo** 파일에 정의된 시스템 서브스크립션을 사용하도록 **osbuild-composer** 서비스를 설정할 수 있습니다. **osbuild-composer** 에서 시스템 서브스크립션을 사용하려면 다음 세부 정보가 있는 리포지토리 덮어쓰기를 정의합니다.

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

**참고**

**osbuild-composer** 는 **/etc/yum.repos.d/** 에 정의된 리포지토리를 자동으로 사용하지 않습니다. 수동으로 시스템 리포지토리 덮어쓰기로 지정하거나 **composer-cli** 를 사용하여 추가 **소스로** 지정해야 합니다. "BaseOS" 및 "AppStream" 리포지토리는 일반적으로 시스템 리포지토리 덮어쓰기를 사용하지만 다른 모든 리포지토리는 **composer-cli** 소스를 사용합니다.

**사전 요구 사항**

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

**절차**

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

```
# cat /etc/yum.repos.d/redhat.repo
[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
      }
   ]
}
```

3. **osbuild-composer.service** 를 다시 시작합니다.

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

**추가 리소스**

- 호스트가 Satellite 6에 등록된 경우 RHEL 이미지 빌더에서 CDN 리포지토리 사용

## 3.7. SATELLITE CV를 콘텐츠 소스로 구성 및 사용

Satellite의 콘텐츠 뷰(CV)를 리포지토리로 사용하여 RHEL 이미지 빌더로 이미지를 빌드할 수 있습니다. 이를 위해 Satellite에 등록된 호스트에서 Red Hat CDN(Content Delivery Network) 공식 리포지토리 대신 Satellite 리포지토리에서 검색할 수 있도록 리포지토리 참조를 수동으로 구성합니다.

**사전 요구 사항**

- Satellite 6에 등록된 호스트에서 RHEL 이미지 빌더를 사용하고 있습니다.

**절차**

1. 현재 구성된 리포지토리에서 리포지토리 URL을 찾습니다.

```
$ sudo yum -v repolist rhel-8-for-x86_64-baseos-rpms | grep repo-baseurl
  Repo-baseurl :
```

다음 출력은 예제입니다.

```
https://satellite6.example.com/pulp/content/YourOrg/YourEnv/YourCV/content/dist/rhel8/8/x86_64/baseos/os
```

2. 하드 코딩된 리포지토리를 Satellite Server로 수정합니다.

   a. **0755** 권한이 있는 리포지토리 디렉터리를 생성합니다.

```
$ sudo mkdir -pvm 0755 /etc/osbuild-composer/repositories
```

   b. **/usr/share/osbuild-composer/repositories/*.json** 의 콘텐츠를 생성한 디렉터리로 복사합니다.

```
$ sudo cp /usr/share/osbuild-composer/repositories/*.json /etc/osbuild-composer/repositories/
```

   c. **/content/dist/*** 행을 통해 Satellite URL 및 파일 콘텐츠를 업데이트합니다.

```
$ sudo sed -i -e 's|cdn.redhat.com|satellite6.example.com/pulp/content/YourOrg/YourEnv/YourCV|' /etc/osbuild-composer/repositories/*.json
```

   d. 구성이 올바르게 교체되었는지 확인합니다.

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

3. 서비스를 다시 시작하십시오.

```
$ sudo systemctl restart osbuild-worker@1.service osbuild-composer.service
```

4. Red Hat 이미지 빌더 구성에서 필요한 시스템 리포지토리를 재정의하고 Satellite 리포지토리의 URL을 baseurl로 사용합니다. 시스템 리포지토리 덮어쓰기 를 참조하십시오.

**추가 리소스**

- Satellite에 여러 사용자 지정 리포지토리가 정의되면 RHEL 이미지 빌더가 실패합니다.

## 3.8. RHEL 이미지 빌더에서 이미지를 빌드하는 데 SATELLITE CV를 리포지토리로 사용

Satellite의 content views(CV)를 리포지토리로 사용하여 사용자 정의 이미지를 빌드하도록 RHEL 이미지 빌더를 구성합니다.

**사전 요구 사항**

- RHEL 웹 콘솔과 Satellite를 통합했습니다. Satellite에서 RHEL 웹 콘솔 활성화를 참조하십시오.

**절차**

1. Satellite 웹 UI에서 **콘텐츠 > 제품으로** 이동하여 제품을 선택하고 사용할 리포지토리를 클릭합니다.
2. **게시됨 필드에서 보안 URL(HTTPS)을 검색하고 복사합니다.**
3. **Red Hat 이미지 빌더 리포지토리의 baseurl으로 복사한 URL을 사용합니다. RHEL 이미지 빌더에 사용자 지정 타사 리포지토리 추가 를 참조하십시오.**

**다음 단계**

- **이미지를 빌드합니다. 웹 콘솔 인터페이스에서 RHEL 이미지 빌더를 사용하여 시스템 이미지 생성을 참조하십시오.**

# 4장. RHEL 이미지 빌더 CLI를 사용하여 시스템 이미지 생성

RHEL 이미지 빌더는 사용자 정의 시스템 이미지를 생성하는 툴입니다. RHEL 이미지 빌더를 제어하고 사용자 정의 시스템 이미지를 생성하려면 CLI(명령줄 인터페이스) 또는 웹 콘솔 인터페이스를 사용할 수 있습니다.

## 4.1. RHEL 이미지 빌더 명령줄 인터페이스 소개

RHEL 이미지 빌더 CLI(명령줄 인터페이스)를 사용하여 적절한 옵션 및 하위 명령으로 **composer-cli** 명령을 실행하여 블루프린트를 생성할 수 있습니다.

명령줄 인터페이스의 워크플로는 다음과 같이 요약할 수 있습니다.

1. 기존 블루프린트 정의를 일반 텍스트 파일에 블루프린트 또는 내보내기(*저장*) 생성
2. 텍스트 편집기에서 이 파일을 편집합니다.
3. 블루프린트 텍스트 파일을 다시 이미지 빌더로 가져오기
4. 작성을 실행하여 지침에서 이미지 빌드
5. 다운로드할 이미지 파일을 내보냅니다.

블루프린트를 생성하기 위한 기본 하위 명령 외에도 **composer-cli** 명령은 구성된 블루프린트의 상태를 검사하고 구성합니다.

## 4.2. RHEL 이미지 빌더를 루트가 아닌 사용자로 사용

**composer-cli** 명령을 root가 아닌 것으로 실행하려면 사용자가**weldr** 그룹에 있어야 합니다.

사전 요구 사항

- 사용자를 생성했습니다.

절차

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

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

## 4.3. 명령줄 인터페이스를 사용하여 블루프린트 생성

CLI(명령줄 인터페이스)를 사용하여 새 RHEL 이미지 빌더 블루프린트를 생성할 수 있습니다. 최종 이미지와 해당 사용자 지정(예: 패키지 및 커널 사용자 지정)을 설명합니다.

사전 요구 사항

- root 사용자 또는 welder 그룹의 멤버인 사용자로 로그인했습니다.

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 스키마에 따른 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.

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

- 특정 버전의 경우 8.7.0 과 같은 정확한 버전 번호를 사용하십시오.
- 사용 가능한 최신 버전의 경우 별표 * 를 사용합니다.
- 최신 마이너 버전의 경우 8.*와 같은 형식을 사용하십시오.

3. 필요에 맞게 사용자 정의하십시오. 예를 들어 SMT(Simultaneous Multi Threading)를 비활성화합니다.

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

사용 가능한 추가 사용자 정의는 지원되는 이미지 사용자 지정을 참조하십시오.

4. 예를 들어 파일을 *BLUEPRINT-NAME*.toml로 저장하고 텍스트 편집기를 종료합니다.

5. 다음과 같이 푸시합니다.

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

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

참고

**composer-cli** 를 non-root로 사용하여 이미지를 생성하려면**weldr** 또는 **root** 그룹에 사용자를 추가합니다.

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

검증

- 기존 기능 목록을 나열하여 해당 기능이 푸시되고 있는지 확인합니다.

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

- 방금 추가한 설정을 표시합니다.

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

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

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

RHEL 이미지 빌더에서 사용자 정의 리포지토리에서 패키지의 종속 항목을 해결할 수 없는 경우 **osbuild-composer** 캐시를 제거합니다.

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

추가 리소스

- **osbuild-composer** 는 내 사용자 정의 리포지토리에서 패키지를 해독할 수 없습니다
- 프록시 서버를 사용하여 사용자 지정 RHEL 시스템 이미지 구성

## 4.4. 명령줄 인터페이스를 사용하여 블루프린트 편집

CLI(명령줄) 인터페이스에서 기존 블루프린트를 편집하여 새 패키지를 추가하거나 새 그룹을 정의하고 사용자 지정 이미지를 생성할 수 있습니다. 이를 위해 다음 단계를 수행합니다.

사전 요구 사항

- 블루프린트를 생성했습니다.

절차

1. 기존 블루프린트를 나열합니다.

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

2. 블루프린트를 로컬 텍스트 파일에 저장합니다.

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

3. 텍스트 편집기로 *BLUEPRINT-NAME*.toml 파일을 편집하고 변경합니다.
4. 편집을 완료하기 전에 파일이 유효한인지 확인합니다.
   a. 블루프린트에서 다음 행을 제거합니다.

```
packages = []
```

b. 버전 번호를 0.0.1에서 0.1.0으로 늘립니다. RHEL 이미지 빌더 블루프린트 버전에서는 Semantic Versioning 스키마를 사용해야 합니다. 또한 버전을 변경하지 않으면 패치 버전 구성 요소가 자동으로 증가합니다.

5. 파일을 저장하고 텍스트 편집기를 종료합니다.

6. 블루프린트를 RHEL 이미지 빌더로 다시 푸시합니다.

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

참고

블루프린트를 RHEL 이미지 빌더로 다시 가져오려면 **.toml** 확장을 포함한 파일 이름을 제공하고 다른 명령에서는 블루프린트 이름만 사용합니다.

검증

1. RHEL 이미지 빌더에 업로드된 콘텐츠가 편집 내용과 일치하는지 확인하려면 블루프린트 내용을 나열합니다.

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

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

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

추가 리소스

- 지원되는 이미지 사용자 지정

## 4.5. 명령줄 인터페이스에서 RHEL 이미지 빌더로 시스템 이미지 생성

RHEL 이미지 빌더 명령줄 인터페이스를 사용하여 사용자 지정 RHEL 이미지를 빌드할 수 있습니다. 이를 위해 블루프린트와 이미지 유형을 지정해야 합니다. 선택적으로 배포를 지정할 수도 있습니다. 배포를 지정하지 않으면 호스트 시스템과 동일한 배포 및 버전을 사용합니다. 아키텍처는 호스트의 아키텍처와 동일합니다.

사전 요구 사항

- 이미지에 대한 준비가 되어 있는 devfile이 있습니다. 명령줄 인터페이스를 사용하여 RHEL 이미지 빌더 블루프린트 생성을 참조하십시오.

절차

1. 선택 사항: 생성할 수 있는 이미지 형식을 나열합니다.

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

2. 작성을 시작합니다.

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

*BLUEPRINT-NAME* 을 블루프린트 이름으로 바꾸고,*IMAGE-TYPE* 을 이미지 유형으로 바꿉니다. 사용 가능한 값은 **composer-cli compose types** 명령의 출력을 참조하십시오.

compose 프로세스는 백그라운드에서 시작되고 composer Universally Unique Identifier (UUID)를 표시합니다.

3. 이미지 생성을 완료하는 데 최대 10분이 걸릴 수 있습니다.
compose의 상태를 확인하려면 다음을 수행하십시오.

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

완료된 작성에는 FINISHED 상태 값이 표시됩니다. 목록에서 작성을 확인하려면 해당 UUID를 사용합니다.

4. 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.

- 이미지의 메타데이터를 다운로드하여 구성 요소에 대한 메타데이터의 **.tar** 파일을 가져옵니다.

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

- 이미지 로그를 다운로드합니다.

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

이 명령은 이미지 생성을 위한 로그가 포함된 **.tar** 파일을 생성합니다. 로그가 비어 있으면 저널을 확인할 수 있습니다.

- 저널 확인:

```
$ journalctl | grep osbuild
```

- 이미지 매니페스트를 확인합니다.

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

journal에서 *job_UUID*.json을 찾을 수 있습니다.

추가 리소스

- RHEL 이미지 빌더 추적

## 4.6. 기본 RHEL 이미지 빌더 명령줄 명령

RHEL 이미지 빌더 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 파일**BLUEPRINT-NAME.toml**에 저장 (export)

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

제품 상세 정보

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

TOML 형식의 블루프린트 파일을 RHEL 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>
```

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>
```

더 많은 하위 명령 및 옵션 보기

```
# composer-cli help
```

추가 리소스

- *composer-cli*(1) 매뉴얼 페이지

## 4.7. RHEL 이미지 빌더 블루프린트 형식

RHEL 이미지 빌더 블루프린트는 TOML 형식의 일반 텍스트로 사용자에게 제공됩니다.

일반적인 파일의 요소는 다음과 같습니다.

triggermes 메타데이터

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

*BLUEPRINT-NAME* 및 *LONG FORM DESCRIPTION TEXT*필드는 name 및 description의 이름입니다.

*VERSION* 은 Semantic Versioning 스키마에 따른 버전 번호이며 전체 블루프린트 파일에 대해 한 번만 제공됩니다.

이미지에 포함할 그룹

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

*group* 항목은 이미지에 설치할 패키지 그룹을 설명합니다. 그룹은 다음 패키지 범주를 사용합니다.

- 필수
- 기본값
- 선택 사항
  *group-name* 은 그룹의 이름입니다(예:anaconda-tools, 위젯, piece 또는 사용자 ). Requires 는 필수 및 기본 패키지를 설치합니다. 선택적 패키지를 선택하는 메커니즘이 없습니다.

이미지에 포함할 패키지

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

-

*package-name* 은 httpd, gdb-doc 또는 coreutils 와 같은 패키지의 이름입니다.

*package-version* 은 사용할 버전입니다. 이 필드는**dnf** 버전 사양을 지원합니다.

- 특정 버전의 경우 8.7.0 과 같은 정확한 버전 번호를 사용하십시오.
- 사용 가능한 최신 버전의 경우 별표 * 를 사용하십시오.
- 최신 마이너 버전의 경우 8.*와 같은 형식을 사용하십시오.

포함할 모든 패키지에 대해 이 블록을 반복합니다.

참고

RHEL 이미지 빌더 툴의 패키지와 모듈 간에 차이가 없습니다. 둘 다 RPM 패키지 종속 항목으로 취급됩니다.

## 4.8. 지원되는 이미지 사용자 정의

다음과 같은 블루프린트에 사용자 지정을 추가하여 이미지를 사용자 지정할 수 있습니다.

- 추가 RPM 패키지 추가
- 서비스 활성화
- 커널 명령줄 매개 변수 사용자 정의.

다른 사람 사이입니다. sshd 내에서 여러 이미지 사용자 지정을 사용할 수 있습니다. 사용자 지정을 사용하면 기본 패키지에서 사용할 수 없는 이미지에 패키지 및 그룹을 추가할 수 있습니다. 이러한 옵션을 사용하려면 블루프린트에서 사용자 지정을 구성하고 RHEL 이미지 빌더로 가져오기(push)합니다.

추가 리소스

- 파일 시스템 사용자 지정 "크기"를 추가한 후ECDHE 가져오기가 실패합니다

### 4.8.1. 배포 선택

**distro** 필드를 사용하여 이미지를 구성할 때 사용할 배포를 선택하거나 블루프린트의 종속성을 해결할 수 있습니다. **distro** 를 비워 두면 호스트 배포를 사용합니다. 배포를 지정하지 않으면 블루프린트에서 호스트 배포를 사용합니다. 호스트 운영 체제를 업그레이드하는 경우 새 운영 체제 버전을 사용하여 배포 세트 빌드 이미지가 없는 블루프린트입니다. RHEL 이미지 빌더 호스트와 다른 운영 체제 이미지를 빌드할 수 없습니다.

- 항상 지정된 RHEL 이미지를 빌드하도록 RHEL 배포를 사용하여 블루프린트를 사용자 지정합니다.

```
name = "blueprint_name"
description = "blueprint_version"
version = "0.1"
distro = "different_minor_version"
```

예를 들면 다음과 같습니다.

```
name = "tmux"
description = "tmux image with openssh"
```

```
version = "1.2.16"
distro = "rhel-8.5"
```

"*different_minor_version*"을 교체하여 다른 마이너 버전을 빌드합니다. 예를 들어 RHEL 8.10 이미지를 빌드하려면 **distro** = "rhel-810"을 사용합니다. RHEL 8.10 이미지에서 RHEL 8.9 및 이전 릴리스와 같은 마이너 버전을 빌드할 수 있습니다.

### 4.8.2. 패키지 그룹 선택

패키지 및 모듈로 블루프린트를 사용자 지정합니다. **name** 속성은 필수 문자열입니다. **version** 속성은 제공되지 않는 경우 리포지토리의 최신 버전을 사용하는 선택적 문자열입니다.

참고

현재 **osbuild-composer** 의 패키지와 모듈 간에는 차이가 없습니다. 둘 다 RPM 패키지 종속성으로 취급됩니다.

- 패키지로 블루프린트를 사용자 지정합니다.

```
[[packages]]
name = "package_group_name"
```

"*package_group_name*"을 그룹 이름으로 바꿉니다. 예를 들면 "tmux"입니다.

```
[[packages]]
name = "tmux"
version = "2.9a"
```

### 4.8.3. 컨테이너 포함

블루프린트를 사용자 지정하여 최신 RHEL 컨테이너를 포함할 수 있습니다. 컨테이너 목록에는 소스가 있는 오브젝트와 필요한 경우 **tls-verify** 속성이 포함되어 있습니다.

컨테이너 목록 항목은 이미지에 포함할 컨테이너 이미지를 설명합니다.

- 소스 - 필수 필드입니다. 레지스트리의 컨테이너 이미지에 대한 참조입니다. 이 예에서는 **registry.access.redhat.com** 레지스트리를 사용합니다. 태그 버전을 지정할 수 있습니다. 기본 태그 버전은 latest입니다.
- **name** - 로컬 레지스트리에 있는 컨테이너의 이름입니다.
- **tls-verify** - 부울 필드. tls-verify 부울 필드는 전송 계층 보안을 제어합니다. 기본값은 true입니다.

포함된 컨테이너는 자동으로 시작되지 않습니다. 시작하려면 파일 사용자 지정이 포함된 **systemd** 장치 파일 또는 사각형을 만듭니다.

- **registry.access.redhat.com/ubi9/ubi:latest** 의 컨테이너와 호스트의 컨테이너를 포함하려면 블루프린트에 다음 사용자 지정을 추가합니다.

```
[[containers]]
source = "registry.access.redhat.com/ubi9/ubi:latest"
name =  "local-name"
tls-verify = true
```

```
[[containers]]
source = "localhost/test:latest"
local-storage = true
```

**containers-auth.json** 파일을 사용하여 보호된 컨테이너 리소스에 액세스할 수 있습니다. 컨테이너레지스트리 인증 정보를 참조하십시오.

### 4.8.4. 이미지 호스트 이름 설정

**customization .hostname** 은 최종 이미지 호스트 이름을 구성하는 데 사용할 수 있는 선택적 문자열입니다. 이 사용자 지정은 선택 사항이며 설정하지 않으면 블루프린트에서 기본 호스트 이름을 사용합니다.

- 블루프린트를 사용자 지정하여 호스트 이름을 구성합니다.

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

### 4.8.5. 추가 사용자 지정

이미지에 사용자를 추가하고 선택적으로 SSH 키를 설정합니다. 이 섹션의 모든 필드는 이름을 제외하고 선택 사항입니다.

절차

- 이미지에 사용자를 추가하도록 블루프린트를 사용자 지정합니다.

```
[[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
```

```
[[customizations.user]]
name = "admin"
description = "Administrator account"
password = "$6$CHO2$3rN8eviE2t50lmVyBYihTgVRHcaecmeCk31L..."
key = "PUBLIC SSH KEY"
home = "/srv/widget/"
shell = "/usr/bin/bash"
groups = ["widget", "users", "wheel"]
uid = 1200
gid = 1200
expiredate = 12345
```

GID는 선택 사항이며 이미지에 이미 있어야 합니다. 선택적으로 패키지에서 이를 생성하거나, **[customizations.group]** 항목을 사용하여 GID를 생성합니다.

*PASSWORD-HASH* 를 실제 암호 해시 로 바꿉니다. 암호 해시 를 생성하려면 다음과 같은 명령을 사용합니다.

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

다른 자리 표시자를 적절한 값으로 바꿉니다.

**name** 값을 입력하고 필요하지 않은 행을 생략합니다.

포함할 모든 사용자에 대해 이 블록을 반복합니다.

### 4.8.6. 추가 그룹 지정

결과 시스템 이미지에 대한 그룹을 지정합니다. **name** 및 **gid** 속성은 모두 필수입니다.

- 그룹으로 블루프린트를 사용자 지정합니다.

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

포함할 모든 그룹에 대해 이 블록을 반복합니다. 예를 들면 다음과 같습니다.

```
[[customizations.group]]
name = "widget"
gid = 1130
```

### 4.8.7. 기존 사용자를 위한 SSH 키 설정

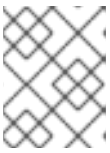
custom **.sshkey** 를 사용하여 최종 이미지에 있는 기존 사용자의 SSH 키를 설정할 수 있습니다사용자 및 키 속성은 모두 필수입니다.

- 기존 사용자의 SSH 키를 설정하여 블루프린트를 사용자 지정합니다.

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

예를 들면 다음과 같습니다.

```
[[customizations.sshkey]]
user = "root"
key = "SSH key for root"
```



참고

기존 사용자에 대한 customization **.sshkey** 사용자 지정만 구성할 수 있습니다. 사용자를 생성하고 SSH 키를 설정하려면 추가 사용자 사용자 지정을 참조하십시오.

### 4.8.8. 커널 인수 추가

부트 로더 커널 명령줄에 인수를 추가할 수 있습니다. 기본적으로 RHEL 이미지 빌더는 기본 커널을 이미지에 빌드합니다. 그러나 블루프린트에서 커널을 구성하여 커널을 사용자 지정할 수 있습니다.

- 커널 부팅 매개변수 옵션을 기본값에 추가합니다.

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

예를 들면 다음과 같습니다.

```
[customizations.kernel]
name = "kernel-debug"
append = "nosmt=force"
```

### 4.8.9. 시간대 및 NTP 설정

블루프린트를 사용자 지정하여 시간대 및 NTP( *Network Time Protocol* )를 구성할 수 있습니다. **timezone** 및 **ntpservers** 속성은 모두 선택적 문자열입니다. 시간대를 사용자 지정하지 않으면 시스템은 UTC( *Universal Time, Coordinated* )를 사용합니다. NTP 서버를 설정하지 않으면 시스템은 기본 배포를 사용합니다.

- 시간대 및 원하는 **ntpservers** 로 블루프린트를 사용자 지정합니다.

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

예를 들면 다음과 같습니다.

```
[customizations.timezone]
timezone = "US/Eastern"
ntpservers = ["0.north-america.pool.ntp.org", "1.north-america.pool.ntp.org"]
```

**참고**

Google Cloud와 같은 일부 이미지 유형에는 이미 NTP 서버가 설정되어 있습니다. 이미지에 선택한 환경에서 NTP 서버를 부팅해야 하므로 재정의할 수 없습니다. 그러나 블루프린트에서 시간대를 사용자 지정할 수 있습니다.

### 4.8.10. 로케일 설정 사용자 정의

결과 시스템 이미지에 대한 로케일 설정을 사용자 지정할 수 있습니다. 언어 및 키보드 속성은 모두 필수입니다. 다른 많은 언어를 추가할 수 있습니다. 첫 번째 언어는 기본 언어이며 다른 언어는 보조 언어입니다.

**절차**

- 로케일 설정을 설정합니다.

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

예를 들면 다음과 같습니다.

```
[customizations.locale]
languages = ["en_US.UTF-8"]
keyboard = "us"
```

- 언어에서 지원하는 값을 나열하려면 다음 명령을 실행합니다.

```
$ localectl list-locales
```

- 키보드에서 지원하는 값을 나열하려면 다음 명령을 실행합니다.

```
$ localectl list-keymaps
```

### 4.8.11. 방화벽 사용자 정의

결과 시스템 이미지에 대한 방화벽을 설정합니다. 기본적으로 방화벽은 **sshd** 와 같이 포트를 명시적으로 활성화하는 서비스를 제외하고 들어오는 연결을 차단합니다.

**[customizations.firewall]** 또는 **[customizations.firewall.services]** 를 사용하지 않으려는 경우 속성을 제거하거나 빈 목록 []으로 설정합니다. 기본 방화벽 설정만 사용하려는 경우 블루프린트에서 사용자 지정을 생략할 수 있습니다.

참고

Google 및 OpenStack 템플릿은 해당 환경의 방화벽을 명시적으로 비활성화합니다. 블루프린트를 설정하여 이 동작을 재정의할 수 없습니다.

절차

- 다음 설정으로 블루프린트를 사용자 지정하여 다른 포트 및 서비스를 엽니다.

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

여기서 port는 열 포트 또는 포트 및 프로토콜 범위를 포함하는 선택적 문자열 목록입니다. **port:protocol** 형식을 사용하여 포트를 구성할 수 있습니다. **portA-portB:protocol** 형식을 사용하여 포트 범위를 구성할 수 있습니다. 예를 들면 다음과 같습니다.

```
[customizations.firewall]
ports = ["22:tcp", "80:tcp", "imap:tcp", "53:tcp", "53:udp", "30000-32767:tcp", "30000-
32767:udp"]
```

숫자 포트 또는 **/etc/services** 의 해당 이름을 사용하여 포트 목록을 활성화하거나 비활성화할 수 있습니다.

- **customization .firewall.service** 섹션에서 활성화 또는 비활성화할 방화벽 서비스를 지정합니다.

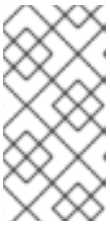
```
[customizations.firewall.services]
enabled = ["SERVICES"]
disabled = ["SERVICES"]
```

- 사용 가능한 방화벽 서비스를 확인할 수 있습니다.

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

예를 들면 다음과 같습니다.

```
[customizations.firewall.services]
enabled = ["ftp", "ntp", "dhcp"]
disabled = ["telnet"]
```



참고

**firewall.services** 에 나열된 서비스는 **/etc/services** 파일에서 사용할 수 있는 서비스 이름과 다릅니다.

### 4.8.12. 서비스 활성화 또는 비활성화

부팅 시 활성화할 서비스를 제어할 수 있습니다. 일부 이미지 유형에는 이미지가 올바르게 작동하고 이 설정을 재정의할 수 없도록 서비스가 이미 활성화되어 있거나 비활성화되어 있습니다. 블루프린트의 **[customizations.services]** 설정은 이러한 서비스를 대체하지 않고 이미지 템플릿에 이미 있는 서비스 목록에 서비스를 추가합니다.

- 부팅 시 활성화할 서비스를 사용자 지정합니다.

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

예를 들면 다음과 같습니다.

```
[customizations.services]
enabled = ["sshd", "cockpit.socket", "httpd"]
disabled = ["postfix", "telnetd"]
```

### 4.8.13. 파티션 모드 지정

**partitioning_mode** 변수를 사용하여 빌드 중인 디스크 이미지를 파티션하는 방법을 선택합니다. 다음과 같은 지원되는 모드로 이미지를 사용자 지정할 수 있습니다.

- **auto-lvm:** 하나 이상의 파일 시스템 사용자 지정이 없는 경우 원시 파티션 모드를 사용합니다. 이 경우 **LVM** 파티션 모드를 사용합니다.

- **LVM:** 추가 마운트 지점이 없는 경우에도 **LVM** 파티션 모드를 사용합니다.

- **Raw:** 마운트 지점이 하나 이상 있는 경우에도 원시 파티션을 사용합니다.

- 다음 사용자 지정을 사용하여 **partitioning_mode** 변수로 블루프린트를 사용자 지정할 수 있습니다.

```
[customizations]
partitioning_mode = "lvm"
```

#### 4.8.14. 사용자 정의 파일 시스템 구성 지정

사용자 정의 파일 시스템 구성을 사용자 정의 파일 시스템 구성을 지정하여 기본 레이아웃 구성 대신 특정 디스크 레이아웃으로 이미지를 생성할 수 있습니다. **vGPU**에 기본이 아닌 레이아웃 구성을 사용하면 다음과 같은 이점을 얻을 수 있습니다.

- 보안 벤치마크 준수

- 디스크 부족 오류로부터 보호

- 성능 개선

- 기존 설정과의 일관성

참고

**OSTree** 이미지에는 읽기 전용과 같은 자체 마운트 규칙이 있으므로 **OSTree** 시스템은 파일 시스템 사용자 정의를 지원하지 않습니다. 다음 이미지 유형은 지원되지 않습니다.

- **image-installer**

- edge-installer
- edge-simplified-installer

또한 이러한 이미지 유형이 분할된 운영 체제 이미지를 생성하지 않기 때문에 다음 이미지 유형은 파일 시스템 사용자 정의를 지원하지 않습니다.

- edge-commit
- edge-container
- tar
- container

RHEL 8.10 및 9.4 이전의 릴리스 배포의 경우 블루프린트는 다음 마운트 지점 및 해당 하위 디렉터리를 지원합니다.

- / - 루트 마운트 지점
- /var
- /home
- /opt
- /srv
- /usr

- /app
- /data
- /tmp

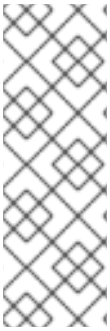
RHEL 9.4 및 8.10 릴리스 이후 릴리스 배포에서는 운영 체제용으로 예약된 특정 경로를 제외하고 임의의 사용자 지정 마운트 지점을 지정할 수 있습니다.

다음 마운트 지점 및 해당 하위 디렉터리에 임의의 사용자 지정 마운트 지점을 지정할 수 없습니다.

- /bin
- /boot/efi
- /dev
- /etc
- /lib
- /lib64
- /lost+found
- /proc
- /run

- /sbin
- /sys
- /sysroot
- /var/lock
- /var/run

/usr 사용자 지정 마운트 지점의 블루프린트에서 파일 시스템을 사용자 지정할 수 있지만 하위 디렉터리는 허용되지 않습니다.



참고

마운트 지점 사용자 지정은 CLI를 사용하여 RHEL 8.5 이후에만 지원됩니다. 이전 배포에서는 root 파티션만 마운트 지점으로 지정하고 크기 인수를 이미지 크기의 별칭으로 지정할 수 있습니다. RHEL 8.6부터 osbuild-composer-46.1-1.el8 RPM 및 이후 버전의 경우 물리적 파티션을 더 이상 사용할 수 없으며 파일 시스템 사용자 지정으로 논리 볼륨을 생성합니다.

사용자 지정 이미지에 파티션이 두 개 이상 있는 경우 LVM에 사용자 지정된 파일 시스템 파티션으로 이미지를 생성하고 런타임 시 해당 파티션의 크기를 조정할 수 있습니다. 이렇게 하려면 블루프린트에서 사용자 지정 파일 시스템 구성을 지정하고 필요한 디스크 레이아웃을 사용하여 이미지를 생성할 수 있습니다. 기본 파일 시스템 레이아웃은 변경되지 않고 남아 있습니다. 파일 시스템 사용자 지정 없이 일반 이미지를 사용하는 경우, cloud-init 는 루트 파티션의 크기를 조정합니다.

블루프린트는 파일 시스템 사용자 지정을 LVM 파티션으로 자동 변환합니다.

사용자 지정 파일을 사용하여 새 파일을 만들거나 기존 파일을 교체할 수 있습니다. 지정한 파일의 상위 디렉터리가 존재해야 합니다. 그렇지 않으면 이미지 빌드가 실패합니다. [customizations.directories] 사용자 지정에 지정하여 상위 디렉터리가 있는지 확인합니다.

주의

파일 사용자 정의를 다른 청사진 사용자 정의와 결합하는 경우 다른 사용자 정의의 기능에 영향을 미치거나 현재 파일 사용자 지정을 재정의할 수 있습니다.

#### 4.8.14.1. 블루프린트에 사용자 지정 파일 지정

**[customizations.files]]** 을 사용하여 다음을 수행할 수 있습니다.

- 새 텍스트 파일을 생성합니다.
- 기존 파일 수정 경고: 기존 콘텐츠를 덮어쓸 수 있습니다.
- 생성 중인 파일의 사용자 및 그룹 소유권을 설정합니다.
- 8진수 형식으로 모드 권한을 설정합니다.

다음 파일을 만들거나 교체할 수 없습니다.

- **/etc/fstab**
- **/etc/shadow**
- **/etc/passwd**
- **/etc/group**

**[customizations.files]** 및 **[[customizations.directories]]** 블루프린트 사용자 지정을 사용하여 이미지에 사용자 지정 파일 및 디렉터리를 생성할 수 있습니다. 이러한 사용자 지정은 **/etc** 디렉토리에서만 사

용할 수 있습니다.

참고

**Edge-raw-image,edge-installer, edge-simplified-installer** 와 같은 **OSTree** 커밋을 배포하는 이미지 유형을 제외하고 모든 이미지 유형에서 이러한 청사진을 지원합니다.

주의

모드,사용자 또는 그룹이 이미 설정된 디렉터리에 이미 존재하는 디렉터리 경로와 함께 **customizations.directories** 를 사용하면 이미지 빌드에서 기존 디렉터리의 소유권 또는 권한을 변경하지 못합니다.

#### 4.8.14.2. 블루프린트에 사용자 지정 디렉터리 지정

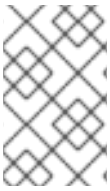
**[customizations.directories]** 를 사용하면 다음을 수행할 수 있습니다.

- 새 디렉터리를 만듭니다.
- 생성 중인 디렉터리에 대한 사용자 및 그룹 소유권을 설정합니다.
- 8진수 형식으로 디렉터리 모드 권한을 설정합니다.
- 필요에 따라 상위 디렉터리가 생성되었는지 확인합니다.

**[customizations.files]]** 을 사용하여 다음을 수행할 수 있습니다.

- 새 텍스트 파일을 생성합니다.
- 기존 파일 수정 경고: 기존 콘텐츠를 덮어쓸 수 있습니다.

- 생성 중인 파일의 사용자 및 그룹 소유권을 설정합니다.
- 8진수 형식으로 모드 권한을 설정합니다.



참고

다음 파일을 만들거나 교체할 수 없습니다.

- /etc/fstab
- /etc/shadow
- /etc/passwd
- /etc/group

다음 사용자 지정을 사용할 수 있습니다.

- 블루프린트에서 파일 시스템 구성을 사용자 지정합니다.

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

**MINIMUM-ECDHEITION-SIZE** 값은 기본 크기 형식이 없습니다. 사용자 지정은 **kB**에서 **TB** 까지, **KiB**~**TiB**의 값 및 단위를 지원합니다. 예를 들어 마운트 지점 크기를 바이트 단위로 정의할 수 있습니다.

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

- 단위를 사용하여 마운트 지점 크기를 정의합니다. 예를 들면 다음과 같습니다.

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

```
[[customizations.filesystem]]
mountpoint = "/boot"
minsize = "1 GiB"
```

- **minsize** 를 설정하여 최소 파티션을 정의합니다. 예를 들면 다음과 같습니다.

```
[[customizations.filesystem]]
mountpoint = "/var"
minsize = 2147483648
```

- **[customizations.directories] :**을 사용하여 이미지의 **/etc** 디렉터리에 사용자 지정 디렉토리를 만듭니다.

```
[[customizations.directories]]
path = "/etc/directory_name"
mode = "octal_access_permission"
user = "user_string_or_integer"
group = "group_string_or_integer"
ensure_parents = boolean
```

해당 항목은 다음과 같이 설명되어 있습니다.

  - **Path - Mandatory -** 생성할 디렉터리의 경로를 입력합니다. 이는 **/etc** 디렉토리 아래의 절대 경로여야 합니다.
  - **mode -** 선택 **-** 디렉터리에 대한 액세스 권한을 **8**진수 형식으로 설정합니다. 권한을 지정하지 않으면 기본값은 **0755**입니다. 선행 **0**은 선택 사항입니다.
  - **user -** 선택 **-** 사용자를 디렉터리의 소유자로 설정합니다. 사용자를 지정하지 않으면 기본값은 **root** 입니다. 사용자를 문자열로 지정하거나 정수로 지정할 수 있습니다.
  - **group -** 선택 사항 **-** 그룹을 디렉터리의 소유자로 설정합니다. 그룹을 지정하지 않으면 기본값은 **root** 입니다. 그룹을 문자열로 지정하거나 정수로 지정할 수 있습니다.
  - **ensure_parents -** 선택 사항 **-** 필요에 따라 상위 디렉터리를 생성할지 여부를 지정합니다. 값을 지정하지 않으면 기본값은 **false** 입니다.

- **[customizations.directories] :**을 사용하여 이미지의 **/etc** 디렉터리에 사용자 지정 파일을 만듭니다.

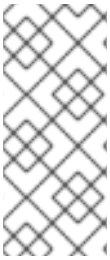
```
[[customizations.files]]
path = "/etc/directory_name"
mode = "octal_access_permission"
user = "user_string_or_integer"
group = "group_string_or_integer"
data = "Hello world!"
```

해당 항목은 다음과 같이 설명되어 있습니다.

  - **Path - Mandatory -** 생성할 파일의 경로를 입력합니다. 이는 **/etc** 디렉토리 아래의 절대 경로여야 합니다.
  - **Mode Optional -** 파일에 대한 액세스 권한을 **8**진수 형식으로 설정합니다. 권한을 지정하지 않으면 기본값은 **0644**입니다. 선행 **0**은 선택 사항입니다.
  - **user -** 선택 **-** 사용자를 파일의 소유자로 설정합니다. 사용자를 지정하지 않으면 기본값은 **root** 입니다. 사용자를 문자열로 지정하거나 정수로 지정할 수 있습니다.
  - **group -** 선택 사항 **-** 그룹을 파일의 소유자로 설정합니다. 그룹을 지정하지 않으면 기본값은 **root** 입니다. 그룹을 문자열로 지정하거나 정수로 지정할 수 있습니다.
  - **Data -** 선택 사항 **-** 일반 텍스트 파일의 내용을 지정합니다. 콘텐츠를 지정하지 않으면 빈 파일이 생성됩니다.

## 4.9. RHEL 이미지 빌더에서 설치한 패키지

**RHEL** 이미지 빌더를 사용하여 시스템 이미지를 생성할 때 시스템은 기본 패키지 그룹 세트를 설치합니다.



참고

구성 요소를 추가할 때 추가한 구성 요소의 패키지가 다른 패키지 구성 요소와 충돌하지 않도록 합니다. 그렇지 않으면 시스템이 종속성을 해결하지 못하고 사용자 지정 이미지를 생성하지 못합니다. 명령을 실행하여 패키지 간에 충돌이 없는지 확인할 수 있습니다.

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

기본적으로 **RHEL** 이미지 빌더에서는 **Core** 그룹을 기본 패키지 목록으로 사용합니다.

표 **4.1.** 이미지 유형 생성을 지원하는 기본 패키지

| 이미지 유형 | 기본 패키지 |
|---|---|
| **ami** | **checkpolicy, chrony, cloud-init, cloud-utils-growpart, @Core, dhcp-client, gdisk, insights-client, kernel, langpacks-en, net-tools, NetworkManager, redhat-release-eECDHE, rng-tools, rsync, selinux-policy-targeted, tar, 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, dnf-plugin-spacewalk, rhn-client-tools, rhnlib, rhn-setup, NetworkManager, dhcp-client, cockpit-ws, cockpit-system, subscription-manager-cockpit, redhat-release-eECDHE, 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-completion, chrony, clevis-dracut, clevis-luks, container-selinux, coreutils, criu, cryptsetup, dnsmasq, dosfstools, dracut-config-generic, dracut-network, e2fsprogs, firewalld, fuse-overlayfs, fwupd, glibc-minimal-langpack, gnupg2, greenboot, gzip, hostname, ima-evm-utils, iproute, iptables, iputils, keyutils, less, lvm2, NetworkManager, NetworkManager-wwan, NetworkManager-wwan, nss-altfiles, OpenSSH-clients, openssh-server, passwd, pinentry, platform-python, policycoreutils, policycoreutils-python-utils, polkit, polkit, redhat-release, rootfiles, rpm, rpm-ostree, rsync, selinux-policy-targeted, setools-console, setup, shadow-utils, shadow-utils, skopeo, slirp4netns, sudo, systemd, tar, tmux, traceroute, usbguard, util-linux, FlexVolume-minimal, wpa_supplicant, xz** |
| **edge-container** | **DNF, dosfstools, e2fsprogs, glibc, lorax-templates-rhel, lvm2, policycoreutils, python3-iniparse, qemu-img, selinux-policy-targeted, systemd, tar, xfsprogs, xz** |

| 이미지 유형 | 기본 패키지 |
| --- | --- |
| **edge-installer** | **aajohan-comfortaa-fonts, badtis-cantarell-fonts, alsa-firmware, alsa-tools-firmware, anaconda, anaconda-install-env-deps, anaconda-widgets, audit, bind-utils-fangsongti-fonts, bzip2, cryptsetup, dbus-x11, dejavu-sans-fonts, device-mapper-persistent-data, dnf, dump, ethtool, ftp, gdb-gdbserver, gdisk, gfs2-utils, glibc-all-langpacks, google-noto-sans-cjk-ttc-fonts, gsettings-ECDHE-schemas, hdparm, hexedit, initscripts, ipmitool, iwl3945-firmware, iwl4965-firmware, iwl4965-firmware, iwl6000g2a-firmware, iwl6000g2b-firmware, jomolhari-fonts, kacst-farsi-fonts, kacst-qurn-fonts, kbd-misc, kbd-misc, khmeros-base-fonts, khmeros-base-fonts, libblockdev-lvm-dbus, libertas-sd8686-firmware, libertas-sd8787-firmware, libertas-usb8388-firmware, libertas-usb8388-olpc-firmware, libibverbs, libreport-plugin-bugzilla libreport-plugin-reportuploader, libreport-rhel-bugzilla, librsvg2, linux-firmware, lklug-fonts, lldpad, lohit-bengali-fonts, lohit-bengali-fonts, lohit-devanagari-fonts, lohit-gujarati-fonts, lohit-gurmukhi-fonts, lohit-kannada-fonts, lohit-odia-fonts, lohit-tamil-fonts, lohit-telugu-fonts, lsof, Madan-fonts, metacity, mtr, mt-st, nmap-ncat, nm-connection-editor, nss-tools, openssh-server, oscap-anaconda-addon, pciutils, perl-interpreter, purez, python3-pyatspi, rdma-core, redhat-release-eECDHE, rpm-ostree, rsync, sg3_utils, sil-abyssinica-fonts, sil-padauk-fonts, sil-scheherazade-fonts, smartmontools, smc-meera-fonts, spice-vdagent, strace, system-storage-manager, thai-scalable-waree-fonts,ECDHEvnc-server-minimal, udisks2, udisks2-iscsi, usbutils, usbutils, volume_key, wget, xfsdump, xorg-x11-drivers, Xorg-x11-fonts-misc,xorg-x11-server-utils,xorg-x11-server-Xorg, xorg-x11-xauth** |

| 이미지 유형 | 기본 패키지 |
| --- | --- |
| edge-simplified-installer | attr, basesystem, binutils, bsdtar, clevis-dracut, clevis-luks, cloud-utils-growpart, coreos-installer-dracut, coreutils, device-mapper-multipath, dnsmasq, dosfstools, dracut-live, e2fsprogs, FCoE-utils, fdo-init, gzip, ima-evm-utils, iproute, iputils, iscsi-initiator-utils, keyutils, lldpad, lvm2, passwd, policycoreutils, policycoreutils-python-utils, procps-ng, setools-console, sudo, traceroute, util-linux |
| image-installer | Anaconda-dracut, curl, dracut-config-generic, dracut-network, 호스트 이름, iwl100-firmware, iwl1000-firmware, iwl105-firmware, iwl135-firmware, iwl2000-firmware, iwl2000-firmware, iwl2030-firmware, iwl2030-firmware, iwl3160-firmware, iwl5000-firmware, iwl5150-firmware, iwl6000-firmware, iwl60-firmware, kernel, less, nfs-utils, openssh-clients, ostree, plymouth, prefixdevname, rng-tools, rpcbind, selinux-policy-targeted, systemd, tar, xfsprogs, xz |
| edge-raw-image | DNF, dosfstools, e2fsprogs, glibc, lorax-templates-rhel, lvm2, policycoreutils, python3-iniparse, qemu-img, selinux-policy-targeted, systemd, tar, xfsprogs, xz |
| gce | @core, langpacks-en, acpid, dhcp-client, dnf-automatic, net-tools, python3, rng-tools, tar,ECDHE |

추가 리소스

- RHEL 이미지 빌더 설명

### 4.10. 사용자 지정 이미지에서 활성화된 서비스

이미지 빌더를 사용하여 사용자 정의 이미지를 구성할 때 이미지에 사용되는 기본 서비스는 다음 사항에 따라 결정됩니다.

- osbuild-composer 유틸리티를 사용하는 RHEL 릴리스

- 이미지 유형

예를 들어 **ami** 이미지 유형을 사용하면 기본적으로 **sshd,chronyd, cloud-init** 서비스를 활성화합니다. 이러한 서비스가 활성화되지 않으면 사용자 정의 이미지가 부팅되지 않습니다.

표 4.2. 이미지 유형 생성을 지원하는 활성화된 서비스

| 이미지 유형 | 기본 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 서비스가 기본적으로 활성화되지 않음 |
| **tar** | 추가 서비스가 기본적으로 활성화되지 않음 |
| **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 사용자 지정은 기본적으로 언급된 이미지 유형에 대해 활성화된 서비스를 재정의하지 않습니다.

추가 리소스

- 지원되는 이미지 사용자 지정

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

RHEL 이미지 빌더는 사용자 정의 시스템 이미지를 생성하는 툴입니다. RHEL 이미지 빌더를 제어하고 사용자 정의 시스템 이미지를 생성하려면 웹 콘솔 인터페이스를 사용할 수 있습니다. 명령줄 인터페이스는 더 많은 기능을 제공하므로 현재 권장되는 대안입니다.

## 5.1. RHEL 웹 콘솔에서 RHEL 이미지 빌더 대시보드에 액세스

RHEL 웹 콘솔용 cockpit-composer 플러그인을 사용하면 이미지 빌더를 관리하고 그래픽 인터페이스를 사용하여 작성할 수 있습니다.

사전 요구 사항

- 시스템에 대한 루트 액세스 권한이 있어야 합니다.
- RHEL 이미지 빌더가 설치되어 있어야 합니다.
- cockpit-composer 패키지를 설치했습니다.

절차

1. 호스트에서 웹 브라우저에서 https://<_localhost_>:9090/ 을 엽니다.
2. root 사용자로 웹 콘솔에 로그인합니다.
3. RHEL 이미지 빌더 컨트롤을 표시하려면 창의 왼쪽 상단에 있는 이미지 빌더 버튼을 클릭합니다.

   RHEL 이미지 빌더 대시보드가 열리고 기존 블루프린트가 나열됩니다(있는 경우).

추가 리소스

- RHEL 8 웹 콘솔을 사용하여 시스템 관리

## 5.2. 웹 콘솔 인터페이스에서 블루프린트 생성

사용자 정의된 **RHEL** 시스템 이미지를 생성하기 전에 **template**을 생성하는 것이 필수 단계입니다. 모든 사용자 지정은 선택 사항입니다.

참고

**Red Hat Enterprise Linux 9.2** 이상 버전과 **Red Hat Enterprise Linux 8.8** 이상 버전에서 이러한 청사진 사용자 지정을 사용할 수 있습니다.

사전 요구 사항

- 브라우저에서 웹 콘솔에서 **RHEL** 이미지 빌더 앱을 열었습니다. **RHEL** 웹 콘솔에서 **RHEL** 이미지 빌더 **GUI** 액세스를 참조하십시오.

절차

1. 오른쪽 상단 모서리에서 **Create ECDHE**을 클릭합니다.

   이름 및 설명에 대한 필드가 있는 대화 상자가 열립니다.

2. 세부 정보 페이지에서 다음을 수행합니다.

   a. 이름을 입력하고 선택적으로 해당 설명을 입력합니다.

   b. 다음을 클릭합니다.

3. 선택 사항: 패키지 페이지에서 다음을 수행합니다.

   a. 사용 가능한 패키지 검색에서 패키지 이름을 입력합니다.

   b. 다음 패키지 필드로 이동하려면 > 버튼을 클릭합니다.

   c. 이전 단계를 반복하여 원하는 만큼 패키지를 검색하고 포함합니다.

d.
다음을 클릭합니다.

참고

이러한 사용자 정의는 달리 지정하지 않는 한 모두 선택 사항입니다.

4.
커널 페이지에서 커널 이름과 명령줄 인수를 입력합니다.

5.
파일 시스템 페이지에서 자동 파티셔닝 사용을 선택하거나 이미지 파일 시스템의 파티션을 수동으로 구성할 수 있습니다. 파티션을 수동으로 구성하려면 다음 단계를 완료합니다.

a.
**Manually configure partitions** 버튼을 클릭합니다.

파티션 구성 섹션이 열리고 **Red Hat** 표준 및 보안 가이드를 기반으로 하는 구성이 표시됩니다.

b.
드롭다운 메뉴에서 파티션을 구성하는 세부 정보를 제공합니다.

i.
마운트 지점 필드의 경우 다음 마운트 지점 유형 옵션 중 하나를 선택합니다.

- **/ -** 루트 마운트 지점
- **/var**
- **/home**
- **/opt**
- **/srv**

- /usr
- /app
- /data
- /tmp
- /usr/local

마운트 지점에 경로를 추가할 수도 있습니다(예: /tmp ). 예를 들어: 접두사로 /tmp 를 추가 경로로 지정하면 /var /tmp가 생성됩니다.

참고

선택한 마운트 지점 유형에 따라 파일 시스템 유형이 **xfs** 로 변경됩니다.

ii. 파일 시스템의 최소 크기 파티션 필드의 경우 필요한 최소 파티션 크기를 입력합니다. **Minimum size**(최소 크기) 드롭다운 메뉴에서 **GiB**,**MiB**, **KiB** 와 같은 일반적인 크기 단위를 사용할 수 있습니다. 기본 단위는 **GiB** 입니다.

참고

최소 크기는 **RHEL** 이미지 빌더가 여전히 파티션 크기를 늘릴 수 있음을 의미합니다. 작업 이미지를 생성할 수 없을 경우 파티션 크기를 늘릴 수 있습니다.

c. 파티션을 추가하려면 파티션 추가 버튼을 클릭합니다. 다음과 같은 오류 메시지가 표시됩니다. 중복 파티션: 각 마운트 지점의 파티션은 하나만 생성할 수 있습니다., 다음을 수행할 수 있습니다.

i. 제거 버튼을 클릭하여 중복된 파티션을 제거합니다.

ii. 생성할 파티션의 새 마운트 지점을 선택합니다.

d. 파티션 구성을 완료한 후 다음을 클릭합니다.

6. 서비스 페이지에서 서비스를 활성화하거나 비활성화할 수 있습니다.

a. 활성화 또는 비활성화할 서비스 이름을 입력하거나 쉼표로, 공백으로 또는 **Enter** 키를 눌러 입력합니다. 다음을 클릭합니다.

7. 방화벽 페이지에서 방화벽 설정을 설정합니다.

a. 포트 및 활성화 또는 비활성화하려는 방화벽 서비스를 입력합니다.

b. 영역 추가 버튼을 클릭하여 각 영역의 방화벽 규칙을 독립적으로 관리합니다. 다음을 클릭합니다.

8. 사용자 페이지에서 단계에 따라 사용자를 추가합니다.

a. 사용자 추가를 클릭합니다.

b. 사용자 이름, 암호, **SSH** 키 를 입력합니다. 서버 관리자 확인란을 클릭하여 사용자를 권한 있는 사용자로 표시할 수도 있습니다. 다음을 클릭합니다.

9. 그룹 페이지에서 다음 단계를 완료하여 그룹을 추가합니다.

a. 그룹 추가 버튼을 클릭합니다.

i. 그룹 이름과 그룹 **ID** 를 입력합니다. 더 많은 그룹을 추가할 수 있습니다. 다음을 클릭합니다.

10. **SSH 키** 페이지에서 키 를 추가합니다.

a. **Add key** 버튼을 클릭합니다.

i. **SSH** 키를 입력합니다.

ii. 사용자 를 입력합니다. 다음을 클릭합니다.

11. 시간대 페이지에서 시간대 설정을 설정합니다.

a. 시간대 필드에 시스템 이미지에 추가할 시간대를 입력합니다. 예를 들어 다음 표준 시간대 형식을 추가합니다. **"US/E disastern".**

시간대를 설정하지 않으면 시스템은 **Universal Time, Coordinated (UTC)**를 기본값으로 사용합니다.

b. **NTP** 서버를 입력합니다. 다음을 클릭합니다.

12. 로컬 페이지에서 다음 단계를 완료합니다.

a. **10.0.0.1** 검색 필드에 시스템 이미지에 추가할 패키지 이름을 입력합니다. 예: **[ "en_US.UTF-8"].**

b. **Languages** 검색 필드에 시스템 이미지에 추가할 패키지 이름을 입력합니다. 예: **"us".** 다음을 클릭합니다.

13. 기타 페이지에서 다음 단계를 완료합니다.

a. **Hostname** 필드에 시스템 이미지에 추가할 호스트 이름을 입력합니다. 호스트 이름을 추가하지 않으면 운영 체제에서 호스트 이름을 결정합니다.

b. **Simplifier** 설치 프로그램 이미지에만 필요합니다. 설치 장치 필드에 시스템 이미지에 유

효한 노드를 입력합니다. 예: dev/sda1. 다음을 클릭합니다.

14. FIDO 이미지를 빌드할 때만 필수 사항: FIDO 장치 온보딩 페이지에서 다음 단계를 완료하십시오.

a. Manufacturing server URL 필드에 다음 정보를 입력합니다.

i. DIUN 공개 키 비보안 필드에 비보안 공개 키를 입력합니다.

ii. DIUN 공개 키 해시 필드에 공개 키 해시를 입력합니다.

iii. DIUN 공개 키 루트 인증서 필드에 공개 키 루트 인증서를 입력합니다. 다음을 클릭합니다.

15. OpenSCAP 페이지에서 다음 단계를 완료합니다.

a. 데이터 스트림 필드에 시스템 이미지에 추가할 datastream 수정 명령을 입력합니다.

b. Profile ID 필드에 시스템 이미지에 추가할 profile_id 보안 프로필을 입력합니다. 다음을 클릭합니다.

16. Ignition 이미지를 빌드할 때만 필수 항목입니다. Ignition 페이지에서 다음 단계를 완료합니다.

a. Firstboot URL 필드에 시스템 이미지에 추가할 패키지 이름을 입력합니다.

b. VMDK 데이터 필드에서 파일을 드래그하거나 업로드합니다. 다음을 클릭합니다.

17. . 검토 페이지에서 청사진에 대한 세부 사항을 검토합니다. 생성을 클릭합니다.

RHEL 이미지 빌더 보기가 열리고 기존 블루프린트가 나열됩니다.

### 5.3. RHEL 이미지 빌더 웹 콘솔 인터페이스에서 블루프린트 가져오기

기존의 서비스를 가져오고 사용할 수 있습니다. 시스템이 모든 종속성을 자동으로 해결합니다.

사전 요구 사항

- 브라우저에서 웹 콘솔에서 **RHEL** 이미지 빌더 앱을 열었습니다.
- **RHEL** 이미지 빌더 웹 콘솔 인터페이스에서 사용하기 위해 가져오려는 블루프린트가 있습니다.

절차

1. **RHEL** 이미지 빌더 대시보드에서 블루프린트 가져오기 를 클릭합니다. 가져오기 마법사 가 열립니다.
2. **Upload( 업로드 )** 필드에서 기존 청사진을 드래그하거나 업로드합니다. **TOML** 또는 **JSON** 형식일 수 있습니다.
3. **Import** 를 클릭합니다. 대시보드에는 가져온 청사진이 나열됩니다.

검증

가져온 청사진을 클릭하면 가져온 모든 사용자 정의가 포함된 대시보드에 액세스할 수 있습니다.

- 가져온 청사진에 대해 선택된 패키지를 확인하려면 패키지 탭으로 이동합니다.
  - 모든 패키지 종속성을 나열하려면 모두 를 클릭합니다. 목록은 검색 가능하며 주문될 수 있습니다.

다음 단계

- 선택 사항: 사용자 정의를 수정하려면 다음을 수행합니다.
  - **Customizations** 대시보드에서 변경할 사용자 지정을 클릭합니다. 선택적으로 사용 가

능한 모든 사용자 지정 옵션으로 이동할 수 있습니다.

추가 리소스

- 웹 콘솔 인터페이스에서 RHEL 이미지 빌더를 사용하여 시스템 이미지를 생성합니다.

## 5.4. RHEL 이미지 빌더 웹 콘솔 인터페이스에서 블루프린트 내보내기

다른 시스템에서 사용자 정의를 사용하도록 template을 내보낼 수 있습니다. TOML 또는 JSON 형식으로 청사진을 내보낼 수 있습니다. 두 형식 모두 CLI에서 작동하며 API 인터페이스에서도 작동합니다.

사전 요구 사항

- 브라우저에서 웹 콘솔에서 RHEL 이미지 빌더 앱을 열었습니다.
- 내보내려는 template이 있습니다.

절차

1. 이미지 빌더 대시보드에서 내보낼 청사진을 선택합니다.
2. Export Blueprint를 클릭합니다. 내보내기 전 마법사 가 열립니다.
3. 내보내기 버튼을 클릭하여 블루프린트를 파일로 다운로드하거나 복사 버튼을 클릭하여 블루프린트를 클립보드에 복사합니다.
   a. 필요한 경우 복사 버튼을 클릭하여 청사진을 복사합니다.

검증

- 텍스트 편집기에서 내보낸 블루프린트를 열어 검사하고 검토합니다.

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

다음 단계를 완료하여 청사진에서 사용자 지정 **RHEL** 시스템 이미지를 생성할 수 있습니다.

사전 요구 사항

- 브라우저에서 웹 콘솔에서 **RHEL** 이미지 빌더 앱을 열었습니다.
- 사용자가 만든 것입니다.

절차

1. **RHEL** 이미지 빌더 대시보드에서 블루프린트 탭을 클릭합니다.
2. **FlexVolume** 테이블에서 이미지를 빌드하려는 경우 해당 이미지를 찾습니다.
3. 선택한 이미지의 오른쪽에서 **Create Image** 를 클릭합니다. 이미지 생성 대화 상자 마법사가 열립니다.
4. 이미지 출력 페이지에서 다음 단계를 완료합니다.
   a. **Select aoctavia** 목록에서 원하는 이미지 유형을 선택합니다.
   b. 이미지 출력 유형 목록에서 원하는 이미지 출력 유형을 선택합니다.

      선택한 이미지 유형에 따라 세부 정보를 추가해야 합니다.
5. 다음을 클릭합니다.
6. 검토 페이지에서 이미지 생성에 대한 세부 사항을 검토하고 이미지 생성을 클릭합니다.

   이미지 빌드가 시작되고 완료하는 데 최대 **20**분이 걸립니다.

검증

이미지 빌드가 완료되면 다음을 수행할 수 있습니다.

- 이미지를 다운로드합니다.
  - RHEL 이미지 빌더 대시보드에서 노드 옵션(밀리초) 메뉴 를 클릭하고 이미지 다운로드를 선택합니다.
- 이미지 로그를 다운로드하여 요소를 검사하고 문제가 있는지 확인합니다.
  - RHEL 이미지 빌더 대시보드에서 노드 옵션(밀리초) 메뉴 를 클릭하고 로그 다운로드를 선택합니다.

# 6장. RHEL 이미지 빌더를 사용하여 다른 릴리스에서 시스템 이미지 생성

RHEL 이미지 빌더를 사용하여 RHEL 8.8 및 RHEL 8.7과 같은 호스트와 다른 여러 RHEL 마이너 릴리스의 이미지를 생성할 수 있습니다. 이를 위해 릴리스 배포 필드가 설정된 소스 시스템 리포지토리를 추가하고 올바른 릴리스 배포 필드 세트를 사용하여 블루프린트를 생성할 수 있습니다.

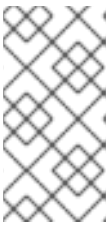
또한 기존 블루프린트 또는 소스 시스템 리포지토리가 이전 형식으로 있는 경우 올바른 릴리스 배포 필드 세트를 사용하여 새 블루프린트를 생성할 수 있습니다.

- 지원되는 릴리스 배포를 나열하려면 다음 명령을 실행합니다.

```
$ composer-cli distros list
```

출력에는 지원되는 릴리스 배포 이름이 있는 목록이 표시됩니다.

```
rhel-8
rhel-84
rhel-85
rhel-86
rhel-87
rhel-88
rhel-89
```



참고

RHEL에서 CentOS 이미지 빌드와 같은 배포 간 이미지 빌드는 지원되지 않습니다.

## 6.1. CLI에서 다른 배포를 사용하여 이미지 생성

RHEL 이미지 빌더 CLI에서 이미지를 구성할 때 사용할 배포를 선택하려면 블루프린트에 distro 필드를 설정해야 합니다. 이를 위해 다음 단계를 수행합니다.

절차

새 블루프린트를 생성하는 경우

1. 블루프린트를 생성합니다. 예를 들면 다음과 같습니다.

```
name = "<blueprint_name>"
description = "<image-description>"
version = "0.0.1"
modules = []
groups = []
distro = "<distro-version>"
```

**distro** 필드를 **"rhel-88"**으로 설정하면 호스트에서 실행 중인 버전과 관계없이 항상 **RHEL 8.8** 이미지를 빌드해야 합니다.

참고

**distro** 필드가 비어 있으면 호스트의 동일한 배포를 사용합니다.

**기존 블루프린트를 업데이트하는 경우**

1. 기존 블루프린트를 로컬 텍스트 파일에 저장**(export)**합니다.

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

1. 선택한 텍스트 편집기를 사용하여 기존 블루프린트 파일을 편집하여 선택한 배포를 사용하여 **distro** 필드를 설정합니다. 예를 들면 다음과 같습니다.

```
name = "blueprint_84"
description = "A 8.8 base image"
version = "0.0.1"
modules = []
groups = []
distro = "rhel-88"
```

2. 파일을 저장하고 편집기를 종료합니다.

3. 블루프린트를 **RHEL** 이미지 빌더로 다시 푸시(가져오기)합니다.

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

4. 이미지 생성을 시작합니다.

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

작성이 완료될 때까지 기다립니다.

5. 구성의 상태를 확인합니다.

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

작성이 완료되면 **FINISHED** 상태 값이 표시됩니다. **UUID**를 통해 목록에서 **compose**를 식별합니다.

6. 결과 이미지 파일을 다운로드합니다.

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

***UUID*** 를 이전 단계에 표시된 **UUID** 값으로 바꿉니다.

## 6.2. 특정 배포판이 있는 시스템 리포지토리 사용

종속성을 해결하고 이미지를 빌드할 때 시스템 리포지토리 소스가 사용하는 배포 문자열 목록을 지정할 수 있습니다. 이를 위해 다음 단계를 수행합니다.

**절차**

- 다음 구조를 사용하여 **TOML** 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
check_gpg = true
check_ssl = true
distros = ["<distro-version>"]
id = "<image-id>"
name = "<image-name>_"
system = false
type = "<image-type>"
url = "\http:///local/repos/rhel-<distro-version>_/<project-repo>/"
```

예를 들면 다음과 같습니다.

```
check_gpg = true
check_ssl = true
```

```
distros = ["rhel-84"]
id = "rhel-84-local"
name = "local packages for rhel-84"
system = false
type = "yum-baseurl"
url = "\http://local/repos/rhel-84/projectrepo/"
```

**추가 리소스**

- 리포지토리 관리.

# 7장. RHEL 이미지 빌더를 사용하여 부팅 ISO 설치 프로그램 이미지 생성

RHEL 이미지 빌더를 사용하여 부팅 가능한 ISO 설치 프로그램 이미지를 생성할 수 있습니다. 이러한 이미지는 루트 파일 시스템이 있는 .tar 파일로 구성됩니다. 부팅 가능한 ISO 이미지를 사용하여 파일 시스템을 베어 메탈 서버에 설치할 수 있습니다.

RHEL 이미지 빌더는 루트 파일 시스템을 포함하는 부팅 ISO를 생성하는 매니페스트를 빌드합니다. ISO 이미지를 생성하려면 이미지 유형 image-installer 를 선택합니다. RHEL 이미지 빌더는 다음 콘텐츠를 사용하여 .tar 파일을 빌드합니다.

- 표준 Anaconda 설치 프로그램 ISO
- 임베디드 RHEL 시스템 tar 파일
- 최소한의 기본 요구 사항으로 커밋을 설치하는 기본 Kickstart 파일

생성된 설치 프로그램 ISO 이미지에는 베어 메탈 서버에 직접 설치할 수 있는 사전 구성된 시스템 이미지가 포함되어 있습니다.

## 7.1. RHEL 이미지 빌더 CLI를 사용하여 부팅 ISO 설치 프로그램 이미지 생성

RHEL 이미지 빌더 명령줄 인터페이스를 사용하여 사용자 지정 부팅 ISO 설치 프로그램 이미지를 생성할 수 있습니다. 결과적으로 이미지 빌더는 운영 체제용으로 설치할 수 있는 .tar 파일이 포함된 .iso 파일을 빌드합니다. .iso 파일은 Anaconda를 부팅하고 .tar 파일을 설치하여 시스템을 설정합니다. 하드 디스크에서 생성된 ISO 이미지 파일을 사용하거나 가상 머신에서 부팅(예: HTTP Boot 또는 USB 설치)할 수 있습니다.

주의

설치 프로그램(.iso) 이미지 유형은 파티션 사용자 지정을 허용하지 않습니다. 파일 시스템 사용자 지정을 수동으로 구성하려고 하면 설치 프로그램 이미지에서 빌드한 모든 시스템에 적용되지 않습니다. RHEL 이미지 빌더 파일 시스템 사용자 지정으로 빌드된 ISO 이미지를 마운트하면 Kickstart에서 오류가 발생하고 설치가 자동으로 재부팅되지 않습니다. 자세한 내용은 이미지 빌더 에서 생성한 RHEL ISO 설치 자동화 및 이미지 빌더에서 생성한 RHEL ISO 설치 자동화를 참조하십시오.

사전 요구 사항

- 이미지에 대한 블루프린트를 생성하고 사용자가 포함된 사용자로 사용자 지정하고 RHEL 이미지 빌더로 다시 푸시했습니다. 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. 작성이 완료될 때까지 기다립니다. 이 작업은 몇 분 정도 걸릴 수 있습니다.

2. 구성의 상태를 확인합니다.

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

   완료된 작성에는 상태 값이 FINISHED로 표시됩니다.

3. UUID를 통해 목록에서 compose를 식별합니다.

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

4. 작성이 완료되면 생성된 이미지 파일을 현재 디렉터리로 다운로드합니다.

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

***UUID***를 이전 단계에서 얻은 **UUID** 값으로 바꿉니다.

**RHEL** 이미지 빌더는 **.tar** 파일이 포함된 **.iso** 파일을 빌드합니다. **.tar** 파일은 운영 체제에 설치될 이미지입니다. **iso**는 **Anaconda**를 부팅하고 **.tar** 파일을 설치하여 시스템을 설정하도록 설정됩니다.

다음 단계

이미지 파일을 다운로드한 디렉터리에서 다음을 수행합니다.

1. 다운로드한 **.iso** 이미지를 찾습니다.

2. **ISO**를 마운트합니다.

```
$ mount -o ro path_to_ISO /mnt
```

**.tar** 파일은 **/mnt/liveimg.tar.gz** 디렉터리에서 찾을 수 있습니다.

3. **.tar** 파일 콘텐츠를 나열합니다.

```
$ tar ztvf /mnt/liveimg.tar.gz
```

추가 리소스

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

## 7.2. GUI에서 RHEL 이미지 빌더를 사용하여 부팅 ISO 설치 프로그램 이미지 생성

RHEL 이미지 빌더 GUI를 사용하여 사용자 지정 부팅 ISO 설치 프로그램 이미지를 빌드할 수 있습니다. 결과 ISO 이미지 파일을 하드 디스크에 사용하거나 가상 시스템에서 부팅할 수 있습니다. 예를 들어 HTTP 부팅 또는 USB 설치에서는 다음과 같습니다.

주의

설치 프로그램(.iso) 이미지 유형은 파티션 사용자 지정을 허용하지 않습니다. 파일 시스템 사용자 지정을 수동으로 구성하려고 하면 설치 프로그램 이미지에서 빌드한 모든 시스템에 적용되지 않습니다. RHEL 이미지 빌더 파일 시스템 사용자 지정으로 빌드된 ISO 이미지를 마운트하면 Kickstart에서 오류가 발생하고 설치가 자동으로 재부팅되지 않습니다. 자세한 내용은 이미지 빌더 에서 생성한 RHEL ISO 설치 자동화 및 이미지 빌더에서 생성한 RHEL ISO 설치 자동화를 참조하십시오.

사전 요구 사항

- 브라우저에서 웹 콘솔에서 RHEL 이미지 빌더 앱을 열었습니다.
- 이미지에 대한 기능을 생성했습니다. 웹 콘솔 인터페이스에서 RHEL 이미지 빌더 블루프린트 생성을 참조하십시오.

절차

1. RHEL 이미지 빌더 대시보드에서 이미지를 빌드하는 데 사용할 블루프린트를 찾습니다. 선택적으로 왼쪽 상단에 있는 검색 상자에 블루프린트 이름 또는 일부를 입력하고 Enter 를 클릭합니다.
2. 오른쪽에 있는 해당 이미지 생성 버튼을 클릭합니다.

   이미지 생성 대화 상자 마법사가 열립니다.
3. 이미지 생성 대화 상자에서 다음을 수행합니다.

a. 이미지 유형 목록에서 **"RHEL Installer(.iso)"** 를 선택합니다.

b. 다음을 클릭합니다.

c. 검토 탭에서 생성 을 클릭합니다.

**RHEL** 이미지 빌더에서는 **RHEL ISO** 이미지 작성을 큐에 추가합니다.

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

검증

성공적으로 생성된 후 이미지를 다운로드할 수 있습니다.

1. 다운로드를 클릭하여 **"RHEL Installer(.iso)"** 이미지를 시스템에 저장합니다.

2. **"RHEL Installer(.iso)"** 이미지를 다운로드한 폴더로 이동합니다.

3. 다운로드한 **.tar** 이미지를 찾습니다.

4. **"RHEL Installer(.iso)"** 이미지 콘텐츠를 추출합니다.

```
$ tar -xf content.tar
```

추가 리소스

- 웹 콘솔 인터페이스에서 **RHEL** 이미지 빌더 블루프린트 생성
- **RHEL** 이미지 빌더 명령줄 인터페이스를 사용하여 시스템 이미지 생성

- RHEL의 부팅 가능 설치 미디어 생성

### 7.3. 미디어에 부팅 가능한 ISO 설치 및 부팅

RHEL 이미지 빌더를 사용하여 베어 메탈 시스템에 생성한 부팅 가능한 ISO 이미지를 설치합니다.

사전 요구 사항

- RHEL 이미지 빌더를 사용하여 부팅 가능한 ISO 이미지를 생성하셨습니다. 명령줄 인터페이스의 RHEL 이미지 빌더를 사용하여 부팅 ISO 설치 프로그램 이미지 생성을 참조하십시오.
- 부팅 가능한 ISO 이미지를 다운로드했습니다.
- dd 도구를 설치했습니다.
- ISO 이미지에 충분한 용량을 갖춘 USB Flash 드라이브가 있습니다. 필요한 크기는 사용자가 선택한 패키지에 따라 달라지지만 권장 최소 크기는 8GB입니다.

절차

1. dd 도구를 사용하여 부팅 가능한 ISO 이미지를 USB 드라이브에 직접 씁니다. 예를 들면 다음과 같습니다.

```
dd if=installer.iso of=/dev/sdX
```

   여기서 installer.iso 는 ISO 이미지 파일 이름이고 /dev/sdX 는 USB Flash 드라이브 장치 경로입니다.

2. 부팅하려는 컴퓨터의 USB 포트에 Flash 드라이브를 삽입합니다.

3. USB 플래시 드라이브에서 ISO 이미지를 부팅합니다.

   설치 환경이 시작되면 기본 Red Hat Enterprise Linux 설치와 유사하게 설치를 수동으로 완료해야 할 수 있습니다.

추가 리소스

- 설치 미디어 부팅
- 설치 사용자 정의
- Linux에서 부팅 가능한 USB 장치 생성

# 8장. RHEL 이미지 빌더 OPENSCAP 통합을 사용하여 사전 강화된 이미지 생성

온프레미스 RHEL 이미지 빌더는 OpenSCAP 통합을 지원합니다. 이러한 통합을 통해 사전 강화된 RHEL 이미지를 생성할 수 있습니다. 블루프린트를 설정하면 다음 작업을 수행할 수 있습니다.

- 사전 정의된 보안 프로필 세트로 사용자 정의
- 패키지 세트 또는 애드온 파일 추가
- 환경에 더 적합한 선택한 플랫폼에 배포할 수 있는 사용자 지정 RHEL 이미지를 빌드합니다.

Red Hat은 현재 배포 지침을 충족할 수 있도록 시스템을 빌드할 때 선택할 수 있는 보안 강화 프로필의 정기적으로 업데이트된 버전을 제공합니다.

## 8.1. KICKSTART와 사전 강화된 이미지 간의 차이점

Kickstart 파일을 사용하여 기존 이미지를 생성하는 경우, 설치 및 시스템에 취약점의 영향을 받지 않도록 해야 하는 패키지를 선택해야 합니다. RHEL 이미지 빌더 OpenSCAP 통합을 사용하면 보안 강화 이미지를 빌드할 수 있습니다. 이미지 빌드 프로세스 중에 OSBuild oscap.remediation 단계는 파일 시스템 트리에서 chroot의 OpenSCAP 툴을 실행합니다. OpenSCAP 툴은 선택한 프로필에 대한 표준 평가를 실행하고 이미지에 수정을 적용합니다. 이를 통해 처음 부팅하기 전에 보안 프로필 요구 사항에 따라 구성할 이미지를 빌드할 수 있습니다.

## 8.2. OPENSCAP 설치

OpenSCAP 툴을 설치하여 시스템의 표준 보안 체크리스트를 생성할 수 있도록 SCAP 툴에 액세스할 수 있습니다.

절차

1. 시스템에 OpenSCAP 을 설치합니다.

```
# *yum install openscap-scanner*
```

2. scap-security-guide 패키지를 설치합니다.

```
# *yum install scap-security-guide*
```

설치가 완료되면 **oscap** 명령줄 툴 사용을 시작할 수 있습니다. 보안 프로필이 있는 **SCAP** 콘텐츠는 **/usr/share/xml/scap/ssg/content/** 디렉터리에 설치됩니다.

## 8.3. OPENSCAP 사용자 정의

블루프린트 사용자 지정에 대한 **OpenSCAP** 지원을 통해 **scap-security-guide** 특정 보안 프로필에 대한 블루프린트를 생성한 다음 이를 사용하여 사전 강화된 이미지를 빌드할 수 있습니다. 사용자 지정 사전 강화된 이미지를 생성하려면 마운트 지점을 수정하고 특정 요구 사항에 따라 파일 시스템 레이아웃을 구성할 수 있습니다. **OpenSCAP** 프로필을 선택한 후 **OpenSCAP** 블루프린트는 선택한 프로필에 따라 이미지 빌드 중에 수정을 트리거하도록 이미지를 구성합니다. 이미지 빌드 중에 **OpenSCAP** 은 사전 부팅 전 수정을 적용합니다.

이미지 블루프린트에서 **OpenSCAP** 블루프린트 사용자 지정을 사용하려면 다음 정보를 제공해야 합니다.

- **datastream** 수정 명령의 데이터 스트림 경로입니다. **scap-security-guide** 패키지의 데이터 스트림 파일은 **/usr/share/xml/scap/ssg/content/** 디렉터리에 있습니다. **1**

- 필요한 보안 프로파일의 **profile_id** 입니다. **profile_id** 필드의 값은 긴 양식과 짧은 양식을 모두 허용합니다. 예를 들어 다음과 같은 양식은 허용됩니다. **cis** 또는 **xccdf_org.ssgproject.content_profile_cis.** 자세한 내용은 **RHEL 8에서 지원되는 SCAP 보안 가이드 프로필** 을 참조하십시오.

  다음 예제는 **OpenSCAP** 수정 단계가 있는 블루프린트입니다.

```
[customizations.openscap]
# If you want to use the data stream from the 'scap-security-guide' package
# the 'datastream' key could be omitted.
# datastream = "/usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml"
profile_id = "xccdf_org.ssgproject.content_profile_cis"
```

명령을 사용하여 제공하는 보안 프로필 목록을 포함하여 **scap-security-guide** 패키지에서 **SCAP** 소스 데이터 스트림에 대한 자세한 내용을 확인할 수 있습니다.

```
# oscap info /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

편의를 위해 **OpenSCAP** 툴은 **scap-security-guide** 데이터 스트림에서 사용 가능한 모든 프로필에 대한 강화 블루프린트를 생성할 수 있습니다.

**예를 들어 명령은**

```
# oscap xccdf generate fix --profile=cis --fix-type=blueprint /usr/share/xml/scap/ssg/content/ssg-rhel8-ds.xml
```

**will generate a 블루프린트 for CIS profile similar to**

```
# Blueprint for CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server
#
# Profile Description:
# This profile defines a baseline that aligns to the "Level 2 - Server"
# configuration from the Center for Internet Security® Red Hat Enterprise
# Linux 8 Benchmark™, v3.0.0, released 2023-10-30.
# This profile includes Center for Internet Security®
# Red Hat Enterprise Linux 8 CIS Benchmarks™ content.
#
# Profile ID:  xccdf_org.ssgproject.content_profile_cis
# Benchmark ID:  xccdf_org.ssgproject.content_benchmark_RHEL-8
# Benchmark Version:  0.1.74
# XCCDF Version:  1.2

name = "hardened_xccdf_org.ssgproject.content_profile_cis"
description = "CIS Red Hat Enterprise Linux 8 Benchmark for Level 2 - Server"
version = "0.1.74"

[customizations.openscap]
profile_id = "xccdf_org.ssgproject.content_profile_cis"
# If your hardening data stream is not part of the 'scap-security-guide' package
# provide the absolute path to it (from the root of the image filesystem).
# datastream = "/usr/share/xml/scap/ssg/content/ssg-xxxxx-ds.xml"

[[customizations.filesystem]]
mountpoint = "/home"
size = 1073741824

[[customizations.filesystem]]
mountpoint = "/tmp"
size = 1073741824

[[customizations.filesystem]]
mountpoint = "/var"
size = 3221225472

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

[[packages]]
name = "aide"
```

```
version = "*"

[[packages]]
name = "libselinux"
version = "*"

[[packages]]
name = "audit"
version = "*"

[customizations.kernel]
append = "audit_backlog_limit=8192 audit=1"

[customizations.services]
enabled = ["auditd","crond","firewalld","systemd-journald","rsyslog"]
disabled = []
masked = ["nfs-server","rpcbind","autofs","bluetooth","nftables"]
```

참고

이미지 강화를 위해 이 정확한 블루프린트 스니펫을 사용하지 마십시오. 이는 전체 프로필을 반영하지 않습니다. **Red Hat**은 **scap-security-guide** 패키지의 각 프로필에 대한 보안 요구 사항을 지속적으로 업데이트하고 구체화하므로 시스템에 제공된 데이터 스트림의 최신 버전을 사용하여 초기 템플릿을 항상 다시 생성하는 것이 좋습니다.

이제 블루프린트를 사용자 지정하거나 그대로 사용하여 이미지를 빌드할 수 있습니다.

**RHEL** 이미지 빌더는 블루프린트 사용자 지정을 기반으로 **osbuild** 단계에 필요한 구성을 생성합니다. 또한 **RHEL** 이미지 빌더에서는 이미지에 두 개의 패키지를 추가합니다.

- **OpenSCAP -scanner - OpenSCAP** 툴.
- **scap-security-guide -** 수정 및 평가 지침이 포함된 패키지입니다.

참고

수정 단계에서는 이 패키지가 기본적으로 이미지에 설치되므로 **datastream**에 **scap-security-guide** 패키지를 사용합니다. 다른 데이터 스트림을 사용하려면 필요한 패키지를"에 추가하고 **oscap** 구성에서 데이터 스트림 경로를 지정합니다.

추가 리소스

- RHEL 8에서 지원되는 SCAP 보안 가이드 프로필.

## 8.4. RHEL 이미지 빌더를 사용하여 사전 강화된 이미지 생성

OpenSCAP 및 RHEL 이미지 빌더 통합을 사용하면 VM에 배포할 수 있는 사전 강화된 이미지를 생성할 수 있습니다.

사전 요구 사항

- root 사용자 또는 welder 그룹의 멤버인 사용자로 로그인되어 있습니다.

절차

1. OpenSCAP 툴 및 scap-security-guide 콘텐츠를 사용하여 TOML 형식으로 강화 블루프린트를 생성하고 필요한 경우 수정합니다.

```
# oscap xccdf generate fix --profile=cis --fix-type=blueprint
/usr/share/xml/scap/ssg/content/ssg-rhel{ProductNumber}-ds.xml > cis.toml
```

2. composer-cli 툴을 사용하여 블루프린트를 osbuild-composer 로 푸시합니다.

```
# composer-cli blueprints push cis.toml
```

3. 강화된 이미지 빌드를 시작합니다.

```
# composer-cli compose start hardened_xccdf_org.ssgproject.content_profile_cis qcow2
```

이미지 빌드가 준비되면 배포에 사전 강화된 이미지를 사용할 수 있습니다. 가상 머신 생성을 참조하십시오.

검증

VM에 사전 강화된 이미지를 배포한 후 구성 규정 준수 검사를 수행하여 이미지가 선택한 보안 프로필에 일치하는지 확인할 수 있습니다.

중요

구성 규정 준수 스캔을 수행해도 시스템이 규정을 준수하는 것은 아닙니다. 자세한 내용은 Configuration compliance scanning 에서 참조하십시오.

1. SSH 를 사용하여 가상 머신에 연결합니다.

2. oscap 스캐너를 실행합니다.

```
# oscap xccdf eval --profile=cis --report=/tmp/compliance-report.html
/usr/share/xml/scap/ssg/content/ssg-rhel{ProductNumber}-ds.xml
```

3. compliance-report.html 을 가져와서 결과를 검사합니다.

추가 리소스

- 구성 규정 준수 및 취약점에 대해 시스템을 스캔합니다.

## 8.5. 프로필의 사용자 지정 맞춤 옵션 블루프린트에 추가

OpenSCAP 및 RHEL 이미지 빌더 통합을 사용하면 다음 옵션을 사용하여 프로필의 사용자 지정 맞춤 옵션을 블루프린트 사용자 지정에 추가할 수 있습니다.

- 추가할 규칙 목록에 대해 선택합니다.
- 제거할 규칙 목록에 대해 선택되지 않음

기본 org.ssgproject.content 규칙 네임스페이스를 사용하면 이 네임스페이스에서 규칙의 접두사를 생략할 수 있습니다. 예를 들어 org.ssgproject.content_grub2_password 및 grub2_password 는 기능적으로 동일합니다.

해당 블루프린트에서 이미지를 빌드할 때 새 맞춤 프로필 ID를 사용하여 맞춤형 파일을 생성하여 이미지에 /usr/share/xml/osbuild-oscap-tailoring/tailoring.xml 으로 저장합니다. 새 프로필 ID에는 기본 프로필에 접미사로 추가된 _osbuild_tailoring 이 있습니다. 예를 들어 CIS(cis) 기본 프로필을 사용하는 경우 프로필 ID는 xccdf_org.ssgproject.content_profile_cis_osbuild_tailoring 이 됩니다.

**사전 요구 사항**

- **root** 사용자 또는 **welder** 그룹의 멤버인 사용자로 로그인되어 있습니다.

**절차**

1. **OpenSCAP** 툴 및 **scap-security-guide** 콘텐츠를 사용하여 **TOML** 형식으로 강화 블루프린트를 생성하고 필요한 경우 수정합니다.

```
# oscap xccdf generate fix --profile=cis --fix-type=blueprint
/usr/share/xml/scap/ssg/content/ssg-rhel{ProductNumber}-ds.xml > cis_tailored.toml
```

2. 사용자 지정 규칙 세트를 사용하여 맞춤형 섹션을 블루프린트에 추가합니다.

```
# Blueprint for CIS Red Hat Enterprise Linux {ProductNumber} Benchmark for Level 2 -
Server
# ...

[customizations.openscap.tailoring]
selected = [ "xccdf_org.ssgproject.content_bind_crypto_policy" ]
unselected = [ "grub2_password" ]
```

3. **composer-cli** 툴을 사용하여 블루프린트를 **osbuild-composer** 로 푸시합니다.

```
# composer-cli blueprints push cis_tailored.toml
```

4. 강화된 이미지 빌드를 시작합니다.

```
# composer-cli compose start hardened_xccdf_org.ssgproject.content_profile_cis qcow2
```

이미지 빌드가 준비되면 배포에서 사전 강화된 이미지를 사용하여 **VM**을 생성합니다. 자세한 내용은 가상 머신 생성을 참조하십시오.

**VM**에 사전 강화된 이미지를 배포한 후 구성 규정 준수 검사를 수행하여 이미지가 선택한 보안 프로필에 일치하는지 확인할 수 있습니다.

중요

구성 규정 준수 스캔을 수행해도 시스템이 규정을 준수하는 것은 아닙니다. 자세한 내용은 Configuration compliance scanning 에서 참조하십시오.

검증

사전 강화된 이미지를 배포한 VM에서 다음 단계를 따르십시오.

1. SSH 를 사용하여 가상 머신에 연결합니다.

2. oscap 스캐너를 실행합니다.

```
# oscap xccdf eval --profile=cis --report=/tmp/compliance-report.html
/usr/share/xml/scap/ssg/content/ssg-rhel{ProductNumber}-ds.xml
```

3. compliance-report.html 을 가져와서 결과를 검사합니다.

추가 리소스

- 구성 규정 준수 및 취약점에 대해 시스템을 스캔합니다.

# 9장. RHEL 이미지 빌더를 사용하여 KVM 게스트 이미지 준비 및 배포

RHEL 이미지 빌더를 사용하여 KVM(커널 기반 가상 시스템) 기반 하이퍼바이저에 배포할 수 있는 .qcow2 용도의 빌드를 생성합니다.

사용자 지정 KVM 게스트 이미지를 생성하려면 다음과 같은 고급 단계를 수행해야 합니다.

1. .qcow2 이미지의 블루프린트를 생성합니다.
2. RHEL 이미지 빌더를 사용하여 .qcow2 이미지를 생성합니다.
3. KVM 게스트 이미지에서 가상 머신을 생성합니다.

## 9.1. RHEL 이미지 빌더를 사용하여 사용자 지정 KVM 게스트 이미지 생성

RHEL 이미지 빌더를 사용하여 사용자 지정 .qcow2 KVM 게스트 이미지를 생성할 수 있습니다. 다음 절차에서는 GUI의 단계를 설명하지만 CLI를 사용할 수도 있습니다.

사전 요구 사항

- 시스템에 액세스하려면 루트 또는 weldr 그룹에 있어야 합니다.
- cockpit-composer 패키지가 설치되어 있습니다.
- RHEL 시스템에서 웹 콘솔의 RHEL 이미지 빌더 대시보드를 열었습니다.
- 네, 네게 만들었어. 웹 콘솔 인터페이스에서 블루프린트 생성을 참조하십시오.

절차

1. 생성한 이름을 클릭합니다.

2. **Images** (이미지) 탭을 선택합니다.

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

4. 유형 드롭다운 메뉴 목록에서 **QEMU Image(.qcow2)** 를 선택합니다.

5. 이미지를 인스턴스화할 때 사용할 크기를 설정하고 생성 을 클릭합니다.

6. 창 오른쪽 상단에 있는 작은 팝업은 이미지 생성이 큐에 추가되었음을 알려줍니다. 이미지 생성 프로세스가 완료되면 이미지 빌드 완료 상태가 표시됩니다.

검증

- 이동 경로 아이콘을 클릭하고 다운로드 옵션을 선택합니다. **RHEL** 이미지 빌더는 기본 다운로드 위치에서 **KVM** 게스트 이미지 **.qcow2** 파일을 다운로드합니다.

추가 리소스

- 웹 콘솔 인터페이스에서 블루프린트 생성.

### 9.2. KVM 게스트 이미지에서 가상 머신 생성

**RHEL** 이미지 빌더를 사용하면 **.qcow2** 이미지를 빌드하고 **KVM** 게스트 이미지를 사용하여 **VM**을 생성할 수 있습니다. **RHEL** 이미지 빌더를 사용하여 생성한 **KVM** 게스트 이미지에는 이미 **cloud-init** 가 설치되어 활성화되어 있습니다.

사전 요구 사항

- **RHEL** 이미지 빌더를 사용하여 **.qcow2** 이미지를 생성했습니다. 웹 콘솔 인터페이스에서 블루프린트 생성을 참조하십시오.

- **qemu-kvm** 패키지가 시스템에 설치되어 있습니다. 시스템에서 **/dev/kvm** 장치를 사용할 수 있는지 확인하고 **BIOS**에서 가상화 기능이 활성화되어 있는지 확인할 수 있습니다.

- 시스템에 **libvirt** 및 **virt-install** 패키지가 설치되어 있어야 합니다.
- 시스템에 설치된 **xorriso** 패키지에서 제공하는 **genisoimage** 유틸리티가 있습니다.

절차

1. **RHEL** 이미지 빌더를 사용하여 생성한 **.qcow2** 이미지를 **/var/lib/libvirt/images/** 디렉터리로 이동합니다.
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
```

**ssh_authorized_keys** 는 **SSH** 공개 키입니다. **SSH** 공개 키는 **~/.ssh/ <*id_rsa.pub*>** 에서 찾을 수 있습니다.

5. **genisoimage** 유틸리티를 사용하여 **user-data** 및 **meta-data** 파일이 포함된 **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-x86_64-kvm.qcow2,device=disk,bus=virtio,format=qcow2 \
    --disk cloud-init.iso,device=cdrom \
    --os-variant rhel 8 \
    --virt-type kvm \
    --graphics none \
    --import
```

- **--graphics none - 헤드리스 RHEL 8 VM임을 나타냅니다.**
- **--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 (Ootpa)
Kernel 4.18.0-221.el8.x86_64 on an x86_64
```

검증

**부팅이 완료되면 VM에 텍스트 로그인 인터페이스가 표시됩니다. VM의 로컬 콘솔에 로그인하려면 user-data 파일의 세부 정보를 사용합니다.**

1. 사용자 이름으로 **admin** 을 입력하고 **Enter** 를 누릅니다.

2. 암호로 **password** 를 입력하고 **Enter** 를 누릅니다.

   로그인 인증이 완료되면 **CLI**를 사용하여 **VM**에 액세스할 수 있습니다.

추가 리소스

- 가상화 활성화.
- **RHEL 8**의 **cloud-init** 구성 및 관리.
- **cloud-init** 중요한 디렉터리 및 파일.

# 10장. 컨테이너를 레지스트리로 푸시하고 이미지에 포함

RHEL 이미지 빌더를 사용하면 OpenSCAP 툴을 사용하여 보안 강화 이미지를 빌드할 수 있습니다. 블루프린트의 컨테이너 사용자 지정 지원을 활용하여 컨테이너를 생성하고 생성한 이미지에 직접 포함할 수 있습니다.

## 10.1. 컨테이너에 이미지에 삽입하기 위한 FLEXVOLUME 사용자 지정

registry.access.redhat.com 레지스트리의 컨테이너를 포함하려면 developer에 컨테이너 사용자 지정을 추가해야 합니다. 예를 들면 다음과 같습니다.

```
[[containers]]
source = "registry.access.redhat.com/ubi9/ubi:latest"
name =  "local-name"
tls-verify = true
```

- 소스 - 필수 필드입니다. 레지스트리의 컨테이너 이미지에 대한 참조입니다. 이 예에서는 registry.access.redhat.com 레지스트리를 사용합니다. 태그 버전을 지정할 수 있습니다. 기본 태그 버전은 latest 입니다.

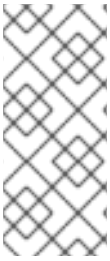
- name - 로컬 레지스트리에 있는 컨테이너의 이름입니다.

- tls-verify - 부울 필드. tls-verify 부울 필드는 전송 계층 보안을 제어합니다. 기본값은 true입니다.

RHEL 이미지 빌더는 이미지 빌드 중에 컨테이너를 가져와서 컨테이너에 이미지에 저장합니다. 기본 로컬 컨테이너 스토리지 위치는 이미지 유형에 따라 달라지므로 Podman과 같은 container-tools 를 모두 지원할 수 있습니다. 포함된 컨테이너는 시작되지 않습니다. 보호된 컨테이너 리소스에 액세스하려면 containers-auth.json 파일을 사용할 수 있습니다.

## 10.2. 컨테이너 레지스트리 인증 정보

osbuild-worker 서비스는 컨테이너 레지스트리와의 통신을 담당합니다. 이를 활성화하려면 /etc/osbuild-worker/osbuild-worker.toml 구성 파일을 설정할 수 있습니다.



참고

/etc/osbuild-worker/osbuild-worker.toml 구성 파일을 설정한 후 **osbuild-worker.toml** 구성 파일을 다시 시작해야 합니다. **osbuild-worker/osbuild-worker.toml** 설정 파일은 **osbuild-worker** 서비스를 시작하는 동안 한 번만 읽기 때문입니다.

**/etc/osbuild-worker/osbuild-worker.toml** 구성 파일에는 보호된 리소스에 액세스하는 데 사용할 **containers-auth.json** 파일의 경로를 참조하는 문자열인 **auth_field_path** 항목이 있는 **containers** 섹션이 있습니다. 컨테이너 레지스트리 인증 정보는 컨테이너를 이미지에 포함할 때 레지스트리에서 컨테이너 이미지를 가져오는 데만 사용됩니다.

예를 들면 다음과 같습니다.

```
[containers]
auth_file_path = "/etc/osbuild-worker/containers-auth.json"
```

추가 리소스

- **containers-auth.json** 도움말 페이지

## 10.3. 컨테이너 레지스트리로 직접 컨테이너 아티팩트 푸시

**RHEL** 이미지 빌더 **CLI**를 사용하여 **RHEL for Edge** 컨테이너 이미지와 같은 컨테이너 아티팩트를 컨테이너 레지스트리로 직접 푸시할 수 있습니다.

사전 요구 사항

- quay.io 레지스트리에 액세스합니다. 이 예에서는 **quay.io** 컨테이너 레지스트리를 대상 레지스트리로 사용하지만 선택한 컨테이너 레지스트리를 사용할 수 있습니다.

절차

1. **registry-config.toml** 파일을 설정하여 컨테이너 공급자를 선택합니다. 인증 정보는 선택 사항입니다.

```
provider = "container_provider"

[settings]
```

```
tls_verify = false
username = "admin"
password = "your_password"
```

2. **.toml 형식으로 설정Setting a .toml format 이는 nginx 패키지를 설치하는 컨테이너의 경우입니다.**

```
name = "simple-container"
description = "Simple RHEL container"
version = "0.0.1"

[[packages]]
name = "nginx"
version = "*"
```

3. **다음과 같이 푸시합니다.**

```
# composer-cli blueprints push blueprint.toml
```

4. **레지스트리와 리포지토리를 인수로 composer-cli 툴에 전달하여 컨테이너 이미지를 빌드합니다.**

```
# composer-cli compose start simple-container container "quay.io:8080/osbuild/repository"
registry-config.toml
```

- **simple-container는 이름이 됩니다.**
- **container - 이미지 유형입니다.**
- **"Quay.io:8080/osbuild/ *repository* " - quay.io 는 대상 레지스트리이고 osbuild 는 조직이며, 빌드가 완료되면 컨테이너를 푸시할 위치입니다. 선택적으로 태그를 설정할 수 있습니다. :tag 의 값을 설정하지 않으면 기본적으로 :latest 태그를 사용합니다.**

**참고**

**사용자 지정 패키지의 종속성을 해결하기 때문에 컨테이너 이미지를 빌드하는 데 시간이 걸립니다.**

5. 이미지 빌드가 완료되면 생성된 컨테이너는 quay.io 에서 사용할 수 있습니다.

검증

1. quay.io 를 열고 리포지토리 태그 를 클릭합니다.

> You can see details about the container you created, such as:
> - last modified
> - image size
> - the `manifest ID`, that you can copy to the clipboard.

2. 매니페스트 ID 값을 복사하여 컨테이너를 포함하려는 이미지를 빌드합니다.

추가 리소스

- quay.io - 태그 작업.

## 10.4. 이미지를 빌드하여 이미지로 컨테이너 가져오기

컨테이너 이미지를 생성한 후에는 사용자 지정 이미지를 빌드하고 컨테이너 이미지를 가져올 수 있습니다. 이를 위해 최종 이미지의 컨테이너 사용자 지정 과 최종 이미지의 컨테이너 이름을 지정해야 합니다. 빌드 프로세스 중에 컨테이너 이미지를 가져와 로컬 Podman 컨테이너 스토리지에 배치됩니다.

사전 요구 사항

- 컨테이너 이미지를 생성하여 로컬 quay.io 컨테이너 레지스트리 인스턴스로 푸시했습니다. 컨테이너 아티팩트를 컨테이너 레지스트리로 직접 푸시를 참조하십시오.
- registry.access.redhat.com 에 액세스할 수 있습니다.
- 컨테이너 매니페스트 ID 가 있습니다.
- qemu-kvm 및 qemu-img 패키지가 설치되어 있어야 합니다.

절차

1.

**qcow2** 이미지를 빌드하기 위해 이름이 생성되었습니다. **template**에는 사용자 지정이 포함되어야 합니다.

```
name = "image"
description = "A qcow2 image with a container"
version = "0.0.1"
distro = "rhel-90"

[[packages]]
name = "podman"
version = "*"

[[containers]]
source = "registry.access.redhat.com/ubi9:8080/osbuild/container/container-image@sha256:manifest-ID-from-Repository-tag: tag-version"
name =  "source-name"
tls-verify = true
```

2. 다음과 같이 푸시합니다.

```
# composer-cli blueprints push blueprint-image.toml
```

3. 컨테이너 이미지를 빌드합니다.

```
# composer-cli start compose image qcow2
```

- ***image*** 는 이름이 됩니다.
- **qcow2** 는 이미지 유형입니다.

참고

이미지를 빌드하는 데 시간이 걸립니다. **quay.io** 레지스트리에서 컨테이너를 확인합니다.

4. **compose**의 상태를 확인하려면 다음을 수행하십시오.

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

완료된 작성에는 **FINISHED** 상태 값이 표시됩니다. 목록에서 작성을 확인하려면 해당 **UUID**

를 사용합니다.

5. 작성 프로세스가 완료되면 결과 이미지 파일을 기본 다운로드 위치로 다운로드합니다.

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

**UUID**를 이전 단계에 표시된 **UUID** 값으로 바꿉니다.

생성하고 다운로드한 **qcow2** 이미지를 사용하여 **VM**을 생성할 수 있습니다.

검증

다운로드한 결과 **qcow2** 이미지에서 다음 단계를 수행합니다.

1. **VM**에서 **qcow2** 이미지를 시작합니다. **KVM** 게스트 이미지에서 가상 머신 생성을 참조하십시오.

2. **qemu** 마법사가 열립니다. **qcow2** 이미지에 로그인합니다.

   a. 사용자 이름과 암호를 입력합니다. 이는 **"customizations.user"** 섹션의 **.qcow2** 에 설정한 사용자 이름과 암호이거나 **cloud-init** 을 사용하여 부팅 시 생성할 수 있습니다.

3. 컨테이너 이미지를 실행하고 컨테이너 내에서 셸 프롬프트를 엽니다.

```
# podman run -it registry.access.redhat.com/ubi9:8080/osbuild/repository /bin/bash/
```

**registry.access.redhat.com** 은 대상 레지스트리이며, **osbuild** 는 조직이며, 빌드가 완료되면 컨테이너를 푸시할 위치입니다.

4. 사용자가 추가한 패키지를 사용할 수 있는지 확인합니다.

```
# type -a nginx
```

출력에 **nginx** 패키지 경로가 표시됩니다.

추가 리소스

- Red Hat 컨테이너 레지스트리 인증.
- Red Hat 레지스트리 액세스 및 구성.
- 기본 Podman 명령.
- 컨테이너에서 Skopeo 실행.

# 11장. RHEL 이미지 빌더를 사용하여 클라우드 이미지 준비 및 업로드

RHEL 이미지 빌더는 다양한 클라우드 플랫폼에서 사용할 준비가 된 사용자 정의 시스템 이미지를 생성할 수 있습니다. 클라우드에서 사용자 지정 RHEL 시스템 이미지를 사용하려면 선택한 출력 유형을 사용하여 RHEL 이미지 빌더로 시스템 이미지를 생성하고, 이미지를 업로드하도록 시스템을 구성하고, 이미지를 클라우드 계정에 업로드합니다. RHEL 웹 콘솔의 이미지 빌더 애플리케이션을 통해 사용자 지정 이미지 클라우드를 푸시할 수 있으며, AWS 및 Microsoft Azure 클라우드와 같이 당사가 지원하는 서비스 공급자 서브 세트에 사용할 수 있습니다. AWS Cloud AMI에 이미지 푸시 및 Microsoft Azure 클라우드에 VHD 이미지 푸시를 참조하십시오.

## 11.1. AWS AMI 이미지 업로드 준비

AWS AMI 이미지를 업로드하기 전에 이미지를 업로드하는 시스템을 구성해야 합니다.

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 python3-pip
```

2. pip 를 사용하여 AWS 명령줄 툴을 설치합니다.

```
# pip3 install awscli
```

3. 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 버킷을 생성합니다.

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

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

5. S3 버킷에 액세스할 수 있는 권한을 부여하려면 AWS IAM(Identity and Access Management)에서 vmimport S3 Role을 생성합니다.

a. 신뢰 정책 구성을 사용하여 JSON 형식으로 trust-policy.json 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
{
   "Version": "2022-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Principal": {
         "Service": "vmie.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
         "StringEquals": {
            "sts:Externalid": "vmimport"
         }
      }
   }]
}
```

b. 역할 정책 구성을 사용하여 JSON 형식으로 role-policy.json 파일을 생성합니다. 예를 들면 다음과 같습니다.

```
{
   "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
```

c. **trust-policy.json** 파일을 사용하여 **Amazon Web Services** 계정에 대한 역할을 생성합니다.

```
$ aws iam create-role --role-name vmimport --assume-role-policy-document file://trust-policy.json
```

d. **role-policy.json** 파일을 사용하여 인라인 정책 문서를 삽입합니다.

```
$ 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

### 11.2. CLI를 사용하여 AWS에 AMI 이미지 업로드

RHEL 이미지 빌더를 사용하여 CLI를 사용하여 ami 이미지를 빌드하고 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 콘텐츠를 사용하여 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 및 리전에 대한 자격 증명으로 바꿉니다. **IMAGE_KEY** 값은 **EC2**에 업로드할 **VM** 이미지의 이름입니다.

2. 파일을 ***CONFIGURATION-FILE*.toml**로 저장하고 텍스트 편집기를 종료합니다.

3. 작성을 시작하여 **AWS**에 업로드합니다.

```
# composer-cli compose start blueprint-name image-type image-key configuration-file.toml
```

**교체:**

- *생성한 블루프린트의 이름이 있는 블루프린트*- 이름
- **ami** *이미지 유형이* 있는 이미지 유형.
- **EC2**에 업로드할 **VM** 이미지의 이름이 있는 이미지 *키* 입니다.
- 클라우드 공급자의 구성 파일 이름이 있는 ***configuration-file*.toml.**

참고

사용자 지정 이미지를 보낼 버킷에 대한 올바른 **AWS IAM(Identity and Access Management)** 설정이 있어야 합니다. 이미지를 업로드하려면 먼저 버킷에 정책을 설정해야 합니다.

4. 이미지 빌드 상태를 확인합니다.

```
# 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. 성공적으로 업로드되었음을 나타내려면 이미지에 사용 가능한 상태가 있어야 합니다.

2. 대시보드에서 이미지를 선택하고 Launch 를 클릭합니다.

추가 리소스

- VM을 가져오는 데 필요한 서비스 역할

### 11.3. AWS CLOUD AMI로 이미지 푸시

RHEL 이미지 빌더를 사용하여 (.raw) 이미지를 생성하고 AWS에 업로드 확인란을 선택하여 생성한 출력 이미지를 Amazon AWS Cloud AMI 서비스 공급자로 직접 푸시할 수 있습니다.

사전 요구 사항

- 시스템에 root 또는 wheel 그룹 사용자가 액세스할 수 있어야 합니다.
- 브라우저에서 RHEL 웹 콘솔의 RHEL 이미지 빌더 인터페이스를 열었습니다.
- 네, 네게 만들었어. 웹 콘솔 인터페이스에서 블루프린트 생성을 참조하십시오.
- AWS IAM 계정 관리자에 대한 액세스 키 ID가 구성되어 있어야 합니다.
- 쓰기 가능한 S3 버킷 이 준비되어 있어야 합니다.

절차

1. **RHEL 이미지 빌더 대시보드에서 이전에 생성한 블루프린트 이름을 클릭합니다.**

2. **Images (이미지) 탭을 선택합니다.**

3. **Create Image (이미지 만들기)를 클릭하여 사용자 지정 이미지를 만듭니다.**

   **이미지 생성 창이 열립니다.**

   a. **유형 드롭다운 메뉴 목록에서 Amazon Machine Image Disk(.raw) 를 선택합니다.**

   b. **Upload to AWS (AWS에 업로드) 확인란을 선택하여 이미지를 AWS Cloud에 업로드하고 Next 를 클릭합니다.**

   c. **AWS에 대한 액세스를 인증하려면 해당 필드에 AWS 액세스 키 ID 및 AWS 시크릿 액세스 키 를 입력합니다. 다음을 클릭합니다.**

      **참고**

      **새 액세스 키 ID를 생성하는 경우에만 AWS 시크릿 액세스 키를 볼 수 있습니다. 보안 키를 모르는 경우 새 액세스 키 ID를 생성합니다.**

   d. **이미지 이름 필드에 이미지 이름을 입력하고 Amazon S3 버킷 이름 필드에 Amazon 버킷 이름을 입력하고 사용자 지정 이미지를 추가할 버킷의 AWS 리전 필드를 입력합니다. 다음을 클릭합니다.**

   e. **정보를 검토하고 마침 을 클릭합니다.**

   **필요한 경우 뒤로 를 클릭하여 잘못된 세부 정보를 수정합니다.**

참고

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

4. 오른쪽 상단에 있는 팝업에 저장 진행 상황을 알려줍니다. 또한 이미지 생성, 이 이미지 생성 진행 및 후속 업로드 **AWS** 클라우드에 대한 이미지 생성이 시작되었음을 알려줍니다.

   프로세스가 완료되면 이미지 빌드 완료 상태가 표시됩니다.

5. 브라우저에서 **Service CryostatEC2** 에 액세스합니다.

   a. **AWS** 콘솔 대시보드 메뉴에서 올바른 리전 을 선택합니다. 이미지가 업로드되었음을 나타내기 위해 **Available** 상태가 있어야 합니다.

   b. **AWS** 대시보드에서 이미지를 선택하고 시작을 클릭합니다.

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

7. 인스턴스 시작 세부 정보를 검토합니다. 변경해야 하는 경우 각 섹션을 편집할 수 있습니다. **Launch**를 클릭합니다.

8. 인스턴스를 시작하기 전에 액세스할 공개 키를 선택합니다.

   이미 보유하고 있는 키 쌍을 사용하거나 새 키 쌍을 만들 수 있습니다.

   다음 단계에 따라 **EC2**에서 새 키 쌍을 생성하고 새 인스턴스에 연결합니다.

   a. 드롭다운 메뉴 목록에서 새 키 쌍 만들기를 선택합니다.

   b.

b. 새 키 쌍의 이름을 입력합니다. 새 키 쌍을 생성합니다.

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

9. 그런 다음 **Launch Instance** (인스턴스 시작)를 클릭하여 인스턴스를 시작할 수 있습니다.

   **Initializing** 으로 표시되는 인스턴스의 상태를 확인할 수 있습니다.

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

11. 연결을 클릭합니다. **SSH**를 사용하여 연결하는 방법에 대한 지침과 함께 창이 표시됩니다.

    a. 에 대한 기본 연결 방법으로 독립 실행형 **SSH** 클라이언트를 선택하고 터미널을 엽니다.

    b. 개인 키를 저장하는 위치에서 **SSH**가 작동하도록 키를 공개적으로 볼 수 있는지 확인합니다. 이렇게 하려면 명령을 실행합니다.

    ```
    $ chmod 400 _your-instance-name.pem_>
    ```

    c. 공용 **DNS**를 사용하여 인스턴스에 연결합니다.

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

    d. **yes** 를 입력하여 연결을 계속할지 확인합니다.

       결과적으로 **SSH**를 통해 인스턴스에 연결됩니다.

검증

- **SSH**를 사용하여 인스턴스에 연결하는 동안 작업을 수행할 수 있는지 확인합니다.

추가 리소스

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

## 11.4. MICROSOFT AZURE VHD 이미지 업로드 준비

RHEL 이미지 빌더를 사용하여 Microsoft Azure 클라우드에 업로드할 수 있는 VHD 이미지를 생성할 수 있습니다.

사전 요구 사항

- Microsoft Azure 리소스 그룹 및 스토리지 계정이 있어야 합니다.
- Python이 설치되어 있어야 합니다. AZ CLI 툴은 python에 따라 다릅니다.

절차

1. Microsoft 리포지토리 키를 가져옵니다.

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

2. 다음 정보를 사용하여 로컬 azure-cli.repo 리포지토리를 생성합니다. azure-cli.repo 리포지토리를 /etc/yum.repos.d/ 아래에 저장합니다.

```
[azure-cli]
name=Azure CLI
baseurl=https://packages.microsoft.com/yumrepos/vscode
enabled=1
gpgcheck=1
gpgkey=https://packages.microsoft.com/keys/microsoft.asc
```

3. Microsoft Azure CLI를 설치합니다.

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

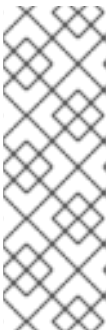
참고

**Microsoft Azure CLI** 패키지의 다운로드된 버전은 현재 사용 가능한 버전에 따라 다를 수 있습니다.

4. **Microsoft Azure CLI**를 실행합니다.

```
$ az login
```

터미널에 다음 메세지가 표시됩니다. 참고로 로그인할 수 있는 브라우저를 시작했습니다. 장치 코드를 사용한 이전 환경의 경우 **"az login --use-device-code** 를 사용하십시오. 그러면 터미널에서 로그인할 수 있는 **https://microsoft.com/devicelogin** 에 대한 링크가 포함된 브라우저를 엽니다.



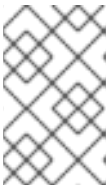
참고

원격(SSH) 세션을 실행하는 경우 브라우저에서 로그인 페이지 링크가 열려 있지 않습니다. 이 경우 브라우저에 대한 링크를 복사하고 원격 세션을 인증하기 위해 로그인할 수 있습니다. 로그인하려면 웹 브라우저를 사용하여 **https://microsoft.com/devicelogin** 페이지를 열고 인증할 장치 코드를 입력합니다.

5. **Microsoft Azure**의 스토리지 계정 키를 나열합니다.

```
$ az storage account keys list --resource-group <resource_group_name> --account-name
<storage_account_name>
```

***resource-group-name*** 을 **Microsoft Azure** 리소스 그룹의 이름으로 바꾸고 ***storage-account-name*** 을 **Microsoft Azure** 스토리지 계정 이름으로 바꿉니다.



참고

다음 명령을 사용하여 사용 가능한 리소스를 나열할 수 있습니다.

```
$ az resource list
```

이전 명령의 출력에서 값 **key1** 을 기록해 둡니다.

1. 스토리지 컨테이너를 생성합니다.

```
$ az storage container create --account-name <storage_account_name>\
--account-key <key1_value> --name <storage_account_name>
```

***storage-account-name*** 을 스토리지 계정 이름으로 교체합니다.

추가 리소스

- Microsoft Azure CLI.

## 11.5. MICROSOFT AZURE 클라우드에 VHD 이미지 업로드

사용자 지정 VHD 이미지를 만든 후에는 Microsoft Azure 클라우드에 업로드할 수 있습니다.

사전 요구 사항

- Microsoft Azure VHD 이미지를 업로드하려면 시스템을 설정해야 합니다. Microsoft Azure VHD 이미지 업로드 준비를 참조하십시오.
- RHEL 이미지 빌더에서 생성한 Microsoft Azure VHD 이미지가 있어야 합니다.
  - CLI에서 vhd 출력 유형을 사용합니다.
  - GUI에서 Azure Disk Image (.vhd) 이미지 유형을 사용합니다.

절차

1. 이미지를 Microsoft Azure에 푸시하고 해당 이미지를 생성합니다.

```
$ az storage blob upload --account-name <account_name> --container-name
<container_name> --file <_image_-disk.vhd> --name <_image_-disk.vhd> --type page
...
```

2. Microsoft Azure Blob 스토리지에 업로드가 완료되면 Microsoft Azure 이미지를 생성합니

다.

```
$ az image create --resource-group &lt; _resource_group_name_&gt; --name &lt;_image_-disk.vhd&gt;  --os-type linux --location eastus --source https://$ACCOUNT.blob.core.windows.net/&lt;_container_name_&gt; &lt;_image_-disk.vhd&gt;
 - Running ...
```

검증

1. Microsoft Azure 포털을 사용하여 인스턴스를 생성하거나 다음과 유사한 명령을 생성합니다.

```
$ az vm create --resource-group &lt;_resource_group_name_&gt; --location eastus --name &lt;_image_-disk.vhd&gt; --image &lt;_image_-disk.vhd&gt; --admin-username azure-user --generate-ssh-keys
 - Running ...
```

2. SSH를 통해 개인 키를 사용하여 결과 인스턴스에 액세스합니다. azure-user 로 로그인합니다. 이 사용자 이름은 이전 단계에서 설정되었습니다.

추가 리소스

- .vhd 형식으로 이미지를 구성하면 실패합니다.

## 11.6. MICROSOFT AZURE 클라우드에 VHD 이미지 푸시

RHEL 이미지 빌더를 사용하여 .vhd 이미지를 생성할 수 있습니다. 그런 다음 출력 .vhd 이미지를 Microsoft Azure Cloud 서비스 공급자의 Blob Storage에 푸시할 수 있습니다.

사전 요구 사항

- 시스템에 대한 루트 액세스 권한이 있어야 합니다.
- RHEL 웹 콘솔의 RHEL 이미지 빌더 인터페이스에 액세스할 수 있습니다.
- 사용자가 만든 것입니다. 웹 콘솔 인터페이스에서 RHEL 이미지 빌더 블루프린트 생성을 참조하십시오.

- Microsoft Storage 계정이 생성되었습니다.
- 쓰기 가능한 Blob 스토리지가 있습니다.

절차

1. RHEL 이미지 빌더 대시보드에서 사용하려는 블루프린트를 선택합니다.
2. 이미지 탭을 클릭합니다.
3. Create Image 를 클릭하여 사용자 지정 .vhd 이미지를 만듭니다.

   이미지 생성 마법사가 열립니다.

   a. 유형 드롭다운 메뉴 목록에서 Microsoft Azure (.vhd) 를 선택합니다.

   b. 이미지를 Microsoft Azure Cloud에 업로드하려면 Azure에 업로드 확인란을 선택합니다.

   c. 이미지 크기를 입력하고 다음을 클릭합니다.

4. Azure에 업로드 페이지에서 다음 정보를 입력합니다.

   a. 인증 페이지에서 다음을 입력합니다.

      i. 스토리지 계정 이름입니다. 스토리지 계정 페이지에서 Microsoft Azure 포털 에서 찾을 수 있습니다.

      ii. 스토리지 액세스 키: 액세스 키 스토리지 페이지에서 찾을 수 있습니다.

iii. 다음을 클릭합니다.

b. 인증 페이지에서 다음을 입력합니다.

i. 이미지 이름입니다.

ii. 스토리지 컨테이너 입니다. 이미지를 업로드할 **Blob** 컨테이너입니다. **Microsoft Azure** 포털에서 **Blob** 서비스 섹션에서 찾습니다.

iii. 다음을 클릭합니다.

5. 검토 페이지에서 생성 을 클릭합니다. **RHEL** 이미지 빌더 및 업로드 프로세스가 시작됩니다.

**Microsoft Azure Cloud** 에 내보낸 이미지에 액세스합니다.

6. **Microsoft Azure** 포털에 액세스합니다.

7. 검색 모음에서 "스토리지 계정"을 입력하고 목록에서 스토리지 계정을 클릭합니다.

8. 검색 모음에서 **"Images"**를 입력하고 **Services** 아래에서 첫 번째 항목을 선택합니다. 이미지 대시보드 로 리디렉션됩니다.

9. 탐색 패널에서 컨테이너를 클릭합니다.

10. 생성한 컨테이너를 찾습니다. 컨테이너 내부에서는 **RHEL** 이미지 빌더를 사용하여 생성하고 푸시한 **.vhd** 파일입니다.

검증

1. **VM** 이미지를 생성하고 시작할 수 있는지 확인합니다.

a. 검색 모음에서 이미지 계정을 입력하고 목록에서 이미지를 클릭합니다.

b. **+Create** 를 클릭합니다.

c. 드롭다운 목록에서 이전에 사용한 리소스 그룹을 선택합니다.

d. 이미지의 이름을 입력합니다.

e. **OS** 유형에 대해 **Linux** 를 선택합니다.

f. **VM** 생성에 대해 **Gen 2** 를 선택합니다.

g. 스토리지 **Blob** 에서 **VHD** 파일에 도달할 때까지 찾아보기 를 클릭하고 스토리지 계정 및 컨테이너를 클릭합니다.

h. 페이지 끝에 있는 **Select** 를 클릭합니다.

i. 계정 유형을 선택합니다(예: **Standard SSD** ).

j. 검토 **+** 생성을 클릭한 다음 생성을 클릭합니다. 이미지 생성을 위해 잠시 기다립니다.

2. **VM**을 시작하려면 단계를 따르십시오.

a. 리소스로 이동을 클릭합니다.

b. 헤더의 메뉴 표시줄에서 **+ Create VM** 을 클릭합니다.

c. 가상 머신의 이름을 입력합니다.

d. 크기 및 관리자 계정 섹션을 완료합니다.

e. 검토 + 생성을 클릭한 다음 생성을 클릭합니다. 배포 진행 상황을 확인할 수 있습니다.

배포가 완료되면 가상 시스템 이름을 클릭하여 SSH를 사용하여 연결할 인스턴스의 공용 IP 주소를 검색합니다.

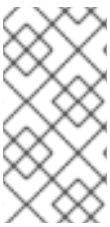
f. 터미널을 열어 VM에 연결할 SSH 연결을 생성합니다.

추가 리소스

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

## 11.7. VMDK 이미지 업로드 및 VSPHERE에서 RHEL 가상 머신 생성

RHEL 이미지 빌더를 사용하면 Open Virtualization 형식(.ova) 또는 가상 디스크(.vmdk) 형식으로 사용자 지정 VMware vSphere 시스템 이미지를 생성할 수 있습니다. 이러한 이미지를 VMware vSphere 클라이언트에 업로드할 수 있습니다. govc import .vmdk CLI 툴을 사용하여 .vmdk 또는 . ova 이미지를 VMware vSphere에 업로드할 수 있습니다. 생성한 vmdk 에는 cloud-init 패키지가 설치되어 있으며 사용자 데이터를 사용하여 사용자를 프로비저닝하는 데 사용할 수 있습니다(예: 사용자 데이터).



참고

VMware vSphere GUI를 사용하여 vmdk 이미지를 업로드하는 것은 지원되지 않습니다.

**사전 요구 사항**

- 사용자 이름 및 암호 사용자 지정으로 **template**을 생성했습니다.
- **RHEL** 이미지 빌더를 사용하여 **.ova** 또는 **.vmdk** 형식으로 **VMware vSphere** 이미지를 생성하고 호스트 시스템에 다운로드합니다.
- **import.vmdk** 명령을 사용할 수 있도록 **govc CLI** 툴을 설치 및 구성하셨습니다.

**절차**

1. **GOVC** 환경 변수를 사용하여 사용자 환경에서 다음 값을 구성합니다.

```
GOVC_URL
GOVC_DATACENTER
GOVC_FOLDER
GOVC_DATASTORE
GOVC_RESOURCE_POOL
GOVC_NETWORK
```

2. **VMware vSphere** 이미지를 다운로드한 디렉터리로 이동합니다.
3. 단계에 따라 **vSphere**에서 **VMware vSphere** 이미지를 시작합니다.

   a. **VMware vSphere** 이미지를 **vSphere**로 가져옵니다.

   ```
   $ govc import.vmdk ./composer-api.vmdk foldername
   ```

   **.ova** 형식의 경우:

   ```
   $ govc import.ova ./composer-api.ova foldername
   ```

   b. 전원을 켜지 않고 **vSphere**에서 **VM**을 생성합니다.

   ```
   govc vm.create \
   -net.adapter=vmxnet3 \
   -m=4096 -c=2 -g=rhel8_64Guest \
   ```

```
-firmware=efi -disk="foldername/composer-api.vmdk" \
-disk.controller=scsi -on=false \
 vmname
```

**.ova** 형식의 경우 **-firmware=efi -disk="*foldername*/composer-api.vmdk"** \ 를 **'-firmware=efi -disk="*foldername*/composer-api.ova"** \로 바꿉니다.

c. **VM**의 전원을 켭니다.

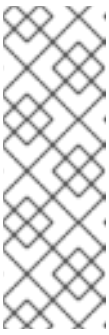
```
govc vm.power -on vmname
```

d. **VM IP** 주소를 검색합니다.

```
govc vm.ip vmname
```

e. **SSH**를 사용하여 **VM**에 로그인합니다. 이때 사용자가 지정한 사용자 이름과 암호를 사용합니다.

```
$ ssh admin@<_ip_address_of_the_vm_>
```



참고

**govc datastore.upload** 명령을 사용하여 로컬 호스트에서 대상으로 **.vmdk** 이미지를 복사한 경우 결과 이미지를 사용할 수 없습니다. **vSphere GUI**에서 **import.vmdk** 명령을 사용할 수 있는 옵션이 없으므로 **vSphere GUI**에서 직접 업로드를 지원하지 않습니다. 결과적으로 **vSphere GUI**에서 **.vmdk** 이미지를 사용할 수 없습니다.

## 11.8. RHEL 이미지 빌더를 사용하여 GCP에 이미지 업로드

**RHEL** 이미지 빌더를 사용하면 **gce** 이미지를 빌드하고 사용자 또는 **GCP** 서비스 계정에 대한 인증 정보를 제공한 다음 **gce** 이미지를 **GCP** 환경에 직접 업로드할 수 있습니다.

### 11.8.1. CLI를 사용하여 GCP에 gce 이미지 업로드

**GCP**에 **gce** 이미지를 업로드하는 인증 정보가 포함된 설정 파일을 설정하려면 절차를 따르십시오.

주의

이미지가 부팅되지 않으므로 **gce** 이미지를 **GCP**에 수동으로 가져올 수 없습니다. **gcloud** 또는 **RHEL** 이미지 빌더를 사용하여 업로드해야 합니다.

사전 요구 사항

- 이미지를 **GCP**에 업로드할 수 있는 유효한 **Google** 계정 및 인증 정보가 있습니다. 자격 증명은 사용자 계정 또는 서비스 계정에서 있을 수 있습니다. 인증 정보와 연결된 계정에는 최소한 다음 **IAM** 역할이 할당되어 있어야 합니다.
  - **roles/storage.admin -** 스토리지 오브젝트 생성 및 삭제
  - **roles/compute.storageAdmin - VM** 이미지를 **Compute Engine**으로 가져오려면 다음을 수행합니다.
- 기존 **GCP** 버킷이 있습니다.

절차

1. 텍스트 편집기를 사용하여 다음 콘텐츠를 사용하여 **gcp-config.toml** 구성 파일을 생성합니다.

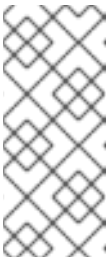
```
provider = "gcp"

[settings]
bucket = "GCP_BUCKET"
region = "GCP_STORAGE_REGION"
object = "OBJECT_KEY"
credentials = "GCP_CREDENTIALS"
```

- **GCP_BUCKET** 은 기존 버킷을 가리킵니다. 업로드 중인 이미지의 중간 스토리지 오브젝트를 저장하는 데 사용됩니다.
- **GCP_STORAGE_REGION** 은 일반 **Google** 스토리지 리전과 듀얼 또는 다중 리전입니다.

- **OBJECT_KEY** 는 중간 스토리지 오브젝트의 이름입니다. 업로드 전에 존재해서는 안 되며 업로드 프로세스가 완료되면 삭제됩니다. 오브젝트 이름이 **.tar.gz** 로 끝나지 않으면 확장 기능이 개체 이름에 자동으로 추가됩니다.

- **GCP_CREDENTIALS** 는 **GCP**에서 다운로드한 인증 정보 **JSON** 파일의 **Base64** 인코딩 체계입니다. 자격 증명은 **GCP**가 이미지를 업로드하는 프로젝트를 결정합니다.



참고

**GCP**로 인증하는 다른 메커니즘을 사용하는 경우 **gcp-config.toml** 파일에 **GCP_CREDENTIALS** 를 지정하는 것은 선택 사항입니다. 기타 인증 방법은 **GCP**로 인증 을 참조하십시오.

2. **GCP**에서 다운로드한 **JSON** 파일에서 **GCP_CREDENTIALS** 를 검색합니다.

```
$ sudo base64 -w 0 cee-gcp-nasa-476a1fa485b7.json
```

3. 추가 이미지 이름 및 클라우드 공급자 프로필을 사용하여 작성을 생성합니다.

```
$ sudo composer-cli compose start BLUEPRINT-NAME gce IMAGE_KEY gcp-config.toml
```

이미지 빌드, 업로드 및 클라우드 등록 프로세스를 완료하는 데 최대 **10**분이 걸릴 수 있습니다.

검증

- 이미지 상태가 **FINISHED**인지 확인합니다.

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

추가 리소스

- **ID** 및 액세스 관리

- 스토리지 버킷 생성

### 11.8.2. RHEL 이미지 빌더가 다른 GCP 인증 정보의 인증 순서를 정렬하는 방법

**RHEL** 이미지 빌더에서 여러 다른 유형의 인증 정보를 사용하여 **GCP**로 인증할 수 있습니다. **RHEL** 이미지 빌더 구성이 여러 인증 정보 세트를 사용하여 **GCP**에서 인증하도록 설정된 경우 다음과 같은 기본 설정 순서대로 인증 정보를 사용합니다.

1. 구성 파일에서 **composer-cli** 명령으로 지정된 자격 증명.
2. **osbuild-composer** 작업자 구성에 구성된 인증 정보.
3. 다음 옵션을 사용하여 자동으로 인증할 수 있는 **Google GCP SDK** 라이브러리의 **Application Default Credentials** (애플리케이션 기본 인증 정보)입니다.
   a. ***GOOGLE_APPLICATION_CREDENTIALS*** 환경 변수가 설정된 경우 애플리케이션 기본 자격 증명은 변수가 가리키는 파일에서 자격 증명을 로드하고 사용합니다.
   b. 애플리케이션 기본 인증 정보는 코드를 실행하는 리소스에 연결된 서비스 계정을 사용하여 인증을 시도합니다. 예를 들면 **Google Compute Engine VM**입니다.

참고

이미지를 업로드할 **GCP** 프로젝트를 확인하려면 **GCP** 인증 정보를 사용해야 합니다. 따라서 모든 이미지를 동일한 **GCP** 프로젝트에 업로드하는 경우가 아니면 **composer-cli** 명령을 사용하여 **gcp-config.toml** 설정 파일에 인증 정보를 지정해야 합니다.

#### 11.8.2.1. composer-cli 명령을 사용하여 GCP 인증 정보 지정

업로드 대상 구성 **gcp-config.toml** 파일에 **GCP** 인증 자격 증명을 지정할 수 있습니다. **Base64**로 인코딩된 **Google** 계정 자격 증명 **JSON** 파일을 사용하여 시간을 절약할 수 있습니다.

절차

1. 다음 명령을 실행하여 **GOOGLE_APPLICATION_CREDENTIALS** 환경 변수에 저장된 경로

를 사용하여 **Google** 계정 자격 증명 파일의 인코딩된 콘텐츠를 가져옵니다.

```
$ base64 -w 0 "${GOOGLE_APPLICATION_CREDENTIALS}"
```

2. 업로드 대상 구성 **gcp-config.toml** 파일에서 인증 정보를 설정합니다.

```
provider = "gcp"

[settings]
provider = "gcp"

[settings]
...
credentials = "GCP_CREDENTIALS"
```

#### 11.8.2.2. osbuild-composer 작업자 구성에서 인증 정보 지정

모든 이미지 빌드에 전역적으로 **GCP**에 사용할 **GCP** 인증 자격 증명을 구성할 수 있습니다. 이렇게 하면 동일한 **GCP** 프로젝트로 이미지를 가져오려면 모든 이미지 업로드에 동일한 인증 정보를 사용할 수 있습니다.

절차

- **/etc/osbuild-worker/osbuild-worker.toml** 작업자 구성에서 다음 인증 정보 값을 설정합니다.

```
[gcp]
credentials = "PATH_TO_GCP_ACCOUNT_CREDENTIALS"
```

## 11.9. RHEL 이미지 빌더 GUI 툴을 사용하여 VMDK 이미지를 VSPHERE로 푸시

이미지 파일을 다운로드하여 수동으로 푸시하지 않도록 **RHEL** 이미지 빌더 **GUI** 툴을 사용하여 **VMware** 이미지를 빌드하고 이미지를 **vSphere** 인스턴스로 직접 푸시할 수 있습니다. 생성한 **vmdk** 에는 **cloud-init** 패키지가 설치되어 있으며 사용자 데이터를 사용하여 사용자를 프로비저닝하는 데 사용할 수 있습니다(예: 사용자 데이터). **RHEL** 이미지 빌더를 사용하여 **.vmdk** 이미지를 빌드하고 **vSphere** 인스턴스 서비스 공급자로 직접 푸시하려면 다음 단계를 따르십시오.

사전 요구 사항

- 루트 또는 **weldr** 그룹의 멤버입니다.

- 브라우저에서 **link:https://localhost:9090/RHEL** 이미지 빌더를 열었습니다.
- 네, 네게 만들었어. 웹 콘솔 인터페이스에서 **RHEL** 이미지 빌더 블루프린트 생성을 참조하십시오.
- vSphere 계정이 있어야 합니다.

절차

1. 생성한 설정의 경우 이미지 탭을 클릭합니다.
2. **Create Image** (이미지 만들기)를 클릭하여 사용자 지정 이미지를 만듭니다.

   **Image type**(이미지 유형) 창이 열립니다.
3. 이미지 유형 창에서 다음을 수행합니다.
   a. 드롭다운 메뉴에서 유형을 선택합니다. **VMware vSphere (.vmdk).**
   b. 이미지를 **vSphere**에 업로드하려면 **Upload to VMware** 확인란을 선택합니다.
   c. 선택 사항: 인스턴스화할 이미지의 크기를 설정합니다. 최소 기본 크기는 **2GB**입니다.
   d. 다음을 클릭합니다.
4. **VMware**에 업로드 창에서 인증 에서 다음 세부 정보를 입력합니다.
   a. 사용자 이름: **vSphere** 계정의 사용자 이름입니다.
   b. **password : vSphere** 계정의 암호입니다.

5.

**VMware에 업로드 창에서 대상 아래에 이미지 업로드 대상에 대한 다음 세부 정보를 입력합니다.**

a.

**image name : 이미지 의 이름입니다.**

b.

**호스트: VMware vSphere의 URL입니다.**

c.

**클러스터: 클러스터의 이름입니다.**

d.

**데이터 센터: 데이터 센터의 이름입니다.**

e.

**데이터 저장소: 데이터 저장소의 이름입니다.**

f.

**다음을 클릭합니다.**

6.

**검토 창에서 이미지 생성 세부 정보를 검토하고 마침 을 클릭합니다.**

**Back 을 클릭하여 잘못된 세부 정보를 수정할 수 있습니다.**

**RHEL 이미지 빌더는 RHEL vSphere 이미지 구성을 큐에 추가하고 지정한 vSphere 인스턴스의 클러스터에 이미지를 생성하고 업로드합니다.**

**참고**

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

**프로세스가 완료되면 이미지 빌드 완료 상태가 표시됩니다.**

검증

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

1. **VMware vSphere Client**에 액세스합니다.

2. 지정한 **vSphere** 인스턴스에서 클러스터에서 이미지를 검색합니다.

3. 업로드한 이미지를 선택합니다.

4. 선택한 이미지를 마우스 오른쪽 버튼으로 클릭합니다.

5. 새 가상 머신을 클릭합니다.

   **New Virtual Machine** 창이 열립니다.

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

   a. 새 가상 머신을 선택합니다.

   b. **VM**의 이름과 폴더를 선택합니다.

   c. 컴퓨터 리소스를 선택합니다. 이 작업에 대한 대상 컴퓨터 리소스를 선택합니다.

   d. 스토리지 선택: 예를 들어, **NFS-Node1**을 선택합니다.

   e. 호환성 선택: 이미지는 **BIOS** 여야 합니다.

   f. 게스트 운영 체제를 선택합니다. 예를 들어 ***Linux*** 및 ***Red Hat Fedora (64비트)*** 를 선택합니다.

   g. 하드웨어 사용자 정의: **VM**을 생성할 때 오른쪽 상단에 있는 장치 구성 버튼에서 기본 새 하드 디스크를 삭제하고 드롭다운을 사용하여 기존 하드 디스크 이미지를 선택합니다.

h. 완료할 준비가 된 경우: 세부 사항을 검토하고 완료 를 클릭하여 이미지를 만듭니다.

6. VM 탭으로 이동합니다.

a. 목록에서 생성한 VM을 선택합니다.

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

c. credential에 대해 생성한 자격 증명을 사용하여 로그인합니다.

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

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

추가 리소스

- vSphere 설치 및 설정 소개

## 11.10. OCI에 사용자 지정 이미지 푸시

RHEL 이미지 빌더를 사용하면 사용자 지정 이미지를 빌드하고 Oracle Cloud Infrastructure(OCI) 인스턴스로 직접 푸시할 수 있습니다. 그런 다음 OCI 대시보드에서 이미지 인스턴스를 시작할 수 있습니다.

사전 요구 사항

- root 또는 weldr 그룹 사용자가 시스템에 액세스할 수 있습니다.
- Oracle Cloud 계정이 있습니다.
- 관리자가 OCI 정책에서 보안 액세스 권한을 부여해야 합니다.

- 선택한 **OCI_REGION** 에 **OCI Bucket**을 생성했습니다.

절차

1. 브라우저에서 웹 콘솔의 **RHEL** 이미지 빌더 인터페이스를 엽니다.
2. 블루프린트 생성을 클릭합니다. 블루프린트 생성 마법사가 열립니다.
3. 세부 정보 페이지에서 블루프린트의 이름을 입력하고 설명을 선택적으로 입력합니다. 다음을 클릭합니다.
4. 패키지 페이지에서 이미지에 포함할 구성 요소 및 패키지를 선택합니다. 다음을 클릭합니다.
5. 사용자 지정 페이지에서 블루프린트에 필요한 사용자 지정을 구성합니다. 다음을 클릭합니다.
6. 검토 페이지에서 생성 을 클릭합니다.
7. 이미지를 생성하려면 이미지 생성을 클릭합니다. 이미지 생성 마법사가 열립니다.
8. 이미지 출력 페이지에서 다음 단계를 완료합니다.
   a. **" Select a blueprint"** 드롭다운 메뉴에서 원하는 블루프린트를 선택합니다.
   b. **"Image output type"** 드롭다운 메뉴에서 **Oracle Cloud Infrastructure (.qcow2)** 를 선택합니다.
   c. **"Upload OCI ( OCI** 업로드) 확인란을 선택하여 이미지를 **OCI**에 업로드합니다.
   d. **"**이미지 크기**"** 를 입력합니다. 다음을 클릭합니다.

9. **OCI -** 인증 업로드 페이지에서 다음 필수 세부 정보를 입력합니다.

   a. 사용자 **OCID:** 사용자 세부 정보를 표시하는 페이지의 콘솔에서 찾을 수 있습니다.

   b. 개인 키

10. **OCI -** 대상 업로드 페이지에서 다음 필수 세부 정보를 입력하고 다음을 클릭합니다.

    a. 이미지 이름**:** 업로드할 이미지의 이름입니다.

    b. **OCI** 버킷

    c. 버킷 네임스페이스

    d. 버킷 리전

    e. 버킷 **compartment**

    f. 버킷 테넌시

11. 마법사의 세부 정보를 검토하고 마침 을 클릭합니다.

**RHEL** 이미지 빌더에서는 **RHEL .qcow2** 이미지 작성을 큐에 추가합니다.

검증

1. **OCI** 대시보드 → 사용자 지정 이미지에 액세스합니다.

2. 이미지에 지정한 **Compartment** 를 선택하고 이미지 가져오기 테이블에서 이미지를 찾습니다.

3. 이미지 이름을 클릭하고 이미지 정보를 확인합니다.

추가 리소스

- OCI에서 사용자 지정 이미지 관리.
- OCI에서 버킷 관리.
- SSH 키 생성.

## 11.11. OPENSTACK에 QCOW2 이미지 업로드

RHEL 이미지 빌더 툴을 사용하면 OpenStack 클라우드 배포에 업로드하는 데 적합한 사용자 지정 .qcow2 이미지를 생성하고 인스턴스를 시작할 수 있습니다. RHEL 이미지 빌더는 QCOW2 형식으로 이미지를 생성하지만 OpenStack에 특정한 변경 사항을 적용합니다.

주의

QCOW2 형식에서도 RHEL 이미지 빌더를 사용하여 생성한 일반 QCOW2 이미지 유형 출력 형식을 잘못 사용하지 말고 OpenStack과 관련된 추가 변경 사항이 포함되어 있습니다.

사전 요구 사항

- 네, 네게 만들었어.

절차

1. QCOW2 이미지 작성을 시작합니다.

```
# composer-cli compose start blueprint_name openstack
```

2. 빌딩 상태를 확인합니다.

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

이미지 빌드가 완료되면 이미지를 다운로드할 수 있습니다.

3. **QCOW2** 이미지를 다운로드합니다.

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

4. **OpenStack** 대시보드에 액세스하여 **+Create Image.**를 클릭합니다.

5. 왼쪽 메뉴에서 **Admin** 탭을 선택합니다.

6. **System Panel** 에서 **Image** 를 클릭합니다.

이미지 생성 마법사가 열립니다.

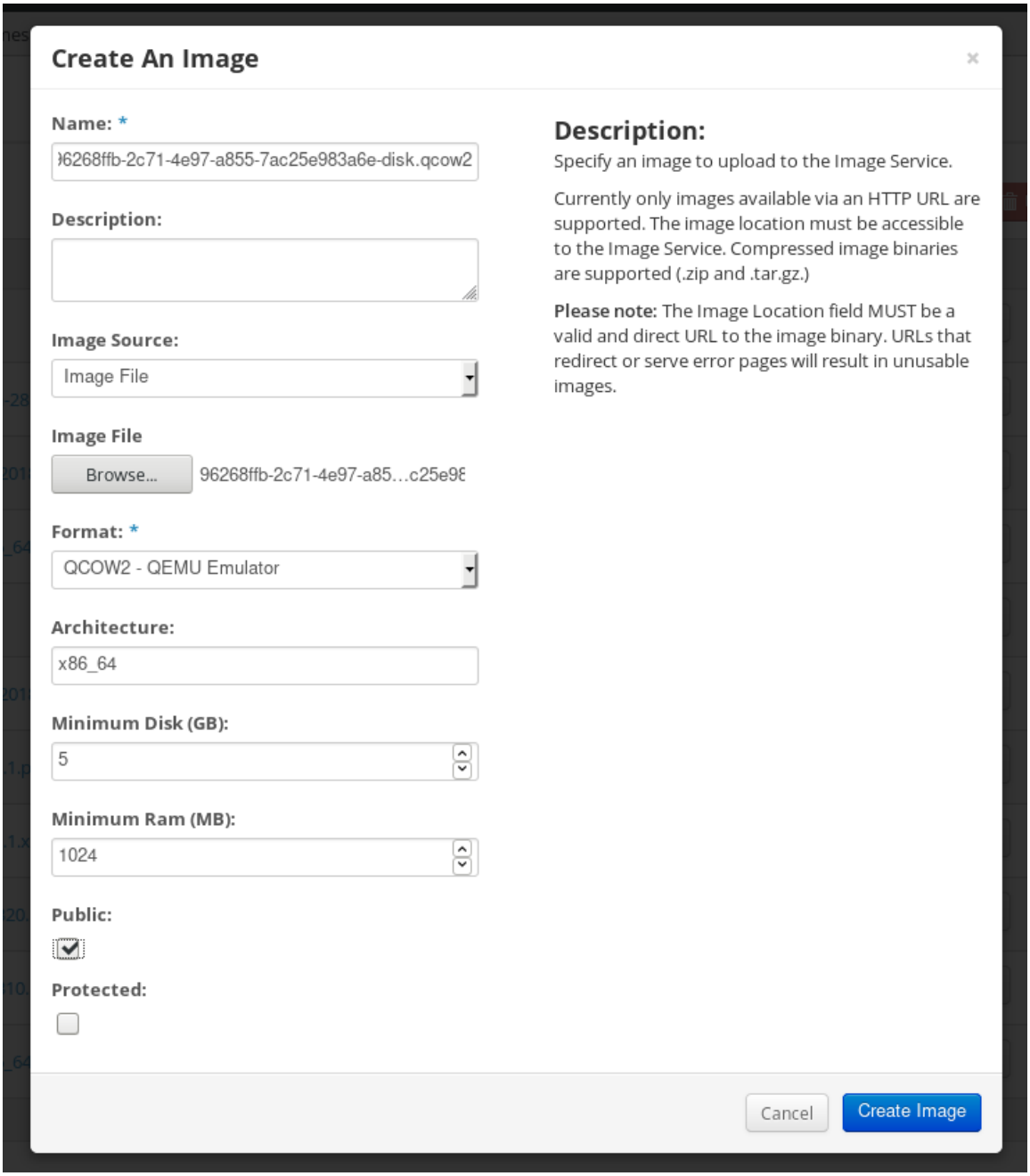
7. 이미지 생성 마법사에서 다음을 수행합니다.

   a. 이미지 이름을 입력합니다.

   b. 찾아보기 를 클릭하여 **QCOW2** 이미지를 업로드합니다.

   c. 형식 드롭다운 목록에서 **QCOW2 - QEMU Emulator** 를 선택합니다.

   d. **Create Image** 를 클릭합니다.




8. 왼쪽 메뉴에서 프로젝트 탭을 선택합니다.

   a. **Compute** 메뉴에서 **Instances** 를 선택합니다.

   b. **Launch Instance** (인스턴스 시작) 버튼을 클릭합니다.

      인스턴스 시작 마법사가 열립니다.

c.

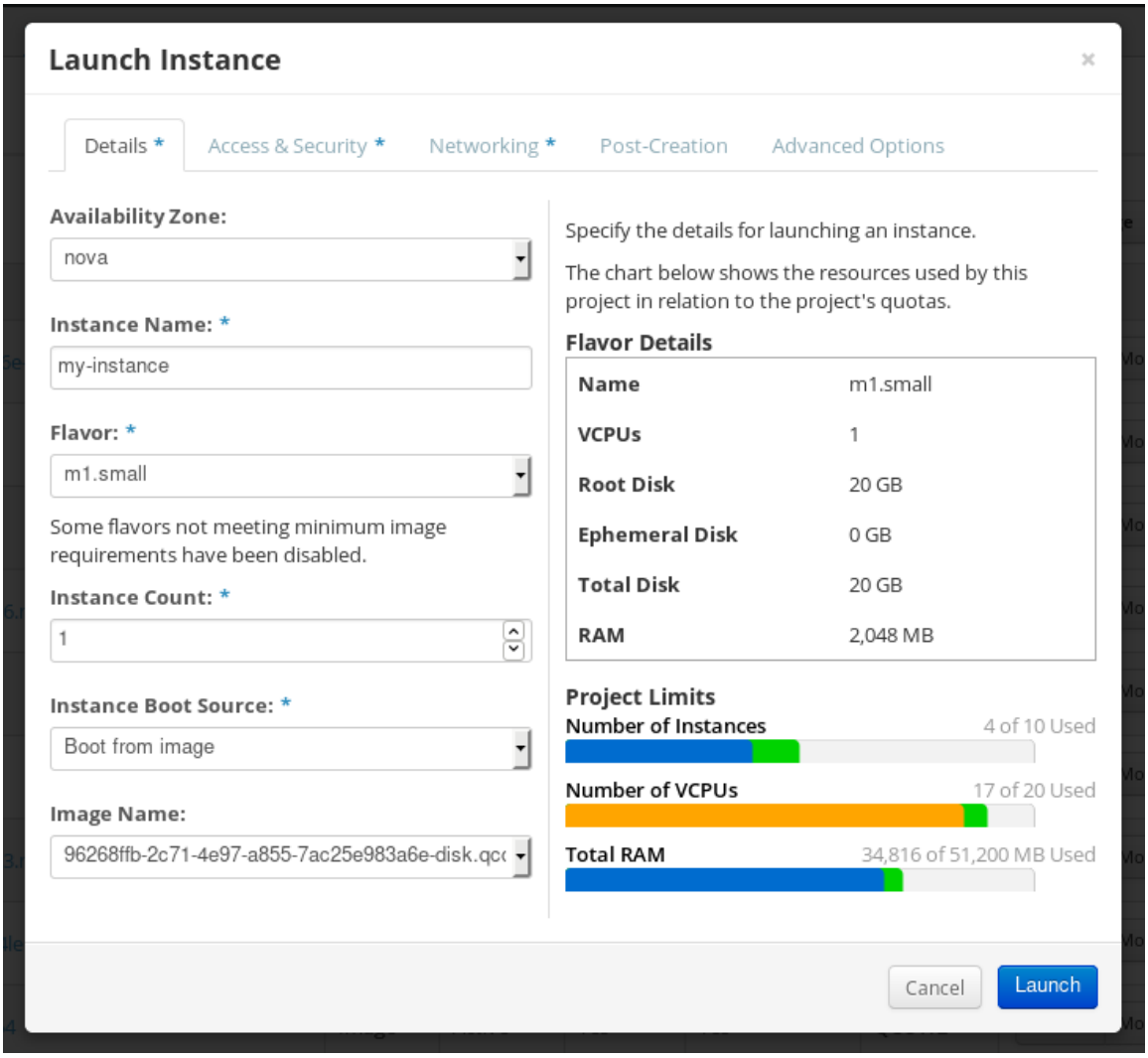
세부 정보 페이지에서 인스턴스의 이름을 입력합니다. 다음을 클릭합니다.

d.

소스 페이지에서 업로드한 이미지의 이름을 선택합니다. 다음을 클릭합니다.

e.

플레이버 페이지에서 필요에 가장 적합한 머신 리소스를 선택합니다. 시작을 클릭합니다.

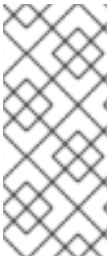



9.

이미지의 메커니즘(CLI 또는 OpenStack 웹 UI)을 사용하여 이미지 인스턴스를 실행할 수 있습니다. SSH를 통해 개인 키를 사용하여 결과 인스턴스에 액세스합니다. cloud-user 로 로그인합니다.

## 11.12. ALIBABA CLOUD에 사용자 지정 RHEL 이미지 업로드 준비

사용자 지정 **RHEL** 이미지를 **Alibaba Cloud**에 배포하려면 먼저 사용자 지정 이미지를 확인해야 합니다. 이미지를 사용하려면 성공적으로 부팅하려면 특정 구성이 필요합니다. **image**를 사용하기 전에 특정 요구 사항을 충족하도록 사용자 지정 이미지를 요청하므로 이미지를 성공적으로 부팅해야 합니다.



참고

**RHEL** 이미지 빌더는 **Alibaba**의 요구 사항을 준수하는 이미지를 생성합니다. 그러나 **Red Hat**은**ECDHE image_check** 툴 을 사용하여 이미지의 형식 준수 여부를 확인하는 것이 좋습니다.

사전 요구 사항

- **RHEL** 이미지 빌더를 사용하여 **Alibaba** 이미지를 생성해야 합니다.

절차

1. **Alibaba image_check** 툴을 사용하여 확인할 이미지가 포함된 시스템에 연결합니다.
2. **image_check** 툴을 다운로드합니다.

```
$ curl -O https://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.

문제 해결

진단 항목 중 실패하는 경우 터미널의 지침에 따라 수정합니다.

추가 리소스

- 이미지 규정 준수 툴.

## 11.13. 사용자 지정 RHEL 이미지를ECDHE에 업로드

RHEL 이미지 빌더를 사용하여 생성한 사용자 지정 AMI 이미지를 Object Storage Service(OSS)에 업로드할 수 있습니다.

사전 요구 사항

- 시스템이 rhcos 이미지를 업로드하도록 설정되어 있습니다. Images Preparing for uploading images를 참조하십시오.
- RHEL 이미지 빌더를 사용하여 ami 이미지를 생성했습니다.
- 버킷이 있습니다. 버킷 생성을 참조하십시오.
- 활성 vGPU 계정이 있습니다.
- OSS 를 활성화했습니다.

절차

1. OSS 콘솔에 로그인합니다.
2. 왼쪽의 Bucket 메뉴에서 이미지를 업로드할 버킷을 선택합니다.
3. 오른쪽 상단 메뉴에서 파일 탭을 클릭합니다.
4. 업로드를 클릭합니다. 오른쪽에 대화 상자가 열립니다. 다음을 구성합니다.

- 업로드 대상: 파일을 현재 디렉터리 또는 지정된 디렉터리에 업로드하도록 선택합니다.
- 파일 ACL: 업로드된 파일의 권한 유형을 선택합니다.

5. 업로드를 클릭합니다.
6. OSS 콘솔에 업로드할 이미지를 선택합니다.
7. 열기를 클릭합니다.

추가 리소스

- 오브젝트를 업로드합니다.
- 사용자 지정 이미지에서 인스턴스 생성.
- 이미지 가져오기.

## 11.14. ALIBABA CLOUD로 이미지 가져오기

RHEL 이미지 빌더를 사용하여 ECS(Elastic Compute Service)로 생성한 사용자 지정 Alibaba RHEL 이미지를 가져오려면 다음 단계를 따르십시오.

사전 요구 사항

- 시스템이 rhcos 이미지를 업로드하도록 설정되어 있습니다. Images Preparing for uploading images를 참조하십시오.
- RHEL 이미지 빌더를 사용하여 ami 이미지를 생성했습니다.
- 버킷이 있습니다. 버킷 생성을 참조하십시오.

- 활성 vGPU 계정이 있습니다.
- OSS 를 활성화했습니다.
- 이미지를 Object Storage Service(OSS)에 업로드했습니다. 이미지 업로드를 참조하십시오.

절차

1. SriovIBNetwork 콘솔에 로그인합니다.
   i. 왼쪽 메뉴에서 이미지를 클릭합니다.
   ii. 오른쪽 상단에서 이미지 가져오기 를 클릭합니다. 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 따라qcow2 또는 ami 입니다.

h. 이미지 설명

i. 데이터 디스크 이미지 추가

이 주소는 OSS 관리 콘솔에서 확인할 수 있습니다. 왼쪽 메뉴에서 필요한 버킷을 선택한 후 다음을 수행합니다.

2. 파일 섹션을 선택합니다.

3. 적절한 이미지에 대한 오른쪽에 있는 세부 정보 링크를 클릭합니다.

화면 오른쪽에 창이 표시되고 이미지 세부 정보가 표시됩니다. OSS 오브젝트 주소는 URL 상자에 있습니다.

4. OK를 클릭합니다.

참고

가져오기 프로세스 시간은 이미지 크기에 따라 달라질 수 있습니다.

사용자 지정 이미지는 ECS 콘솔로 가져옵니다.

추가 리소스

- 이미지 가져오기를 위한 노트입니다.
- 사용자 지정 이미지에서 인스턴스 생성.
- 오브젝트를 업로드합니다.

## 11.15. ALIBABA CLOUD를 사용하여 사용자 지정 RHEL 이미지의 인스턴스 생성

Alibaba ECS 콘솔을 사용하여 사용자 지정 RHEL 이미지의 인스턴스를 생성할 수 있습니다.

사전 요구 사항

- OSS 를 활성화하여 사용자 지정 이미지를 업로드했습니다.
- 이미지를octets Console로 가져왔습니다. 이미지 가져오기를 참조하십시오.

절차

1. SriovIBNetwork 콘솔에 로그인합니다.
2. 왼쪽 메뉴에서 인스턴스를 선택합니다.
3. 오른쪽 상단에서 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.

결과적으로 ECS 콘솔에서 배포할 준비가 된 활성 인스턴스가 있습니다.

추가 리소스

- 사용자 지정 이미지를 사용하여 인스턴스 생성.

- 마법사를 사용하여 인스턴스를 생성합니다.