



U Y U N I

Uyuni 2023.02

클라이언트 구성 가이드

2023년04월18일



차례

클라이언트 구성 안내서 개요	1
1. 지원되는 클라이언트 및 기능	2
1.1. 지원되는 클라이언트 시스템	2
1.2. 지원되는 도구 패키지	3
1.3. 지원하는 SUSE 및 openSUSE 클라이언트 기능	3
1.4. Supported SLE Micro Client Features	6
1.5. openSUSE Leap Micro Client Features	7
1.6. 지원하는 Alibaba Cloud Linux 기능	9
1.7. 지원하는 AlmaLinux 기능	11
1.8. 지원되는 Amazon Linux 기능	13
1.9. 지원하는 CentOS 기능	15
1.10. 지원하는 Debian 기능	17
1.11. 지원하는 Oracle 기능	19
1.12. 지원하는 Red Hat Enterprise Linux 기능	21
1.13. 지원하는 Rocky Linux 기능	23
1.14. 지원하는 Ubuntu 기능	25
2. 구성 기본	28
2.1. 소프트웨어 채널	28
2.1.1. SUSE Package Hub가 제공하는 패키지	28
2.1.2. AppStream이 제공하는 패키지	29
2.1.3. EPEL이 제공하는 패키지	29
2.1.4. SUSE Linux Enterprise 클라이언트의 Unified Installer 업데이트 채널	29
2.1.5. 소프트웨어 리포지토리	29
2.1.6. 소프트웨어 제품	31
2.2. 부트스트랩 리포지토리	31
2.2.1. 부트스트랩 리포지토리 생성 준비	31
2.2.2. 자동 모드를 위한 옵션	32
2.2.3. 부트스트랩 리포지토리를 수동으로 생성	32
2.2.4. 부트스트랩 및 사용자 지정 채널	34
2.3. 활성화 키	34
2.3.1. 여러 활성화 키 결합	36
2.3.2. 활성화 키	36
2.3.3. 활성화 키 모범 사례	37
2.4. GPG 키	38
2.4.1. 클라이언트의 GPG 키 신뢰	38
3. 클라이언트 관리 방법	41
3.1. Salt 클라이언트를 위한 연결 방법	41
3.1.1. 온보딩 정보	41
3.1.2. Salt SSH를 통한 푸시	42
3.1.3. Salt Bundle	44
3.2. 기존 클라이언트를 위한 연결 방법	46
3.2.1. SUSE Manager 데몬(rhnsd)	47
3.2.2. SSH를 통한 푸시	50
4. 클라이언트 등록	55
4.1. 클라이언트 등록 방법	55
4.1.1. Web UI로 클라이언트 등록	55
4.1.2. 부트스트랩 스크립트로 클라이언트 등록	57
4.1.3. 명령줄에서 등록(Salt)	60
4.2. SUSE 클라이언트 등록	62
4.2.1. SUSE Linux Enterprise 클라이언트 등록	63
4.2.2. SLE Micro 클라이언트 등록	66
4.3. openSUSE 클라이언트 등록	70
4.3.1. openSUSE Leap 클라이언트 등록	70
4.3.2. Registering openSUSE Leap Micro Clients	72
4.4. Alibaba Cloud Linux 클라이언트 등록	74
4.4.1. Alibaba Cloud Linux 클라이언트 등록	74
4.5. AlmaLinux 클라이언트 등록	76

4.5.1. AlmaLinux 클라이언트 등록	76
4.6. Amazon Linux 클라이언트 등록.....	78
4.6.1. Amazon Linux 클라이언트 등록	79
4.7. CentOS 클라이언트 등록.....	80
4.7.1. CentOS 클라이언트 등록	81
4.8. Debian 클라이언트등록	85
4.8.1. Debian 클라이언트 등록.....	85
4.9. Oracle 클라이언트 등록.....	88
4.9.1. Oracle Linux 클라이언트 등록.....	88
4.10. Red Hat 클라이언트 등록	90
4.10.1. Red Hat Enterprise Linux 클라이언트를 CDN으로 등록	91
4.10.2. cobbler distro list	182
4.10.3. cobbler profile list	183
4.10.4. mount -o loop,ro <image_name>.iso /mnt	185
4.10.5. mkdir -p /srv/www/distributions.....	186
4.10.6. cp -a /mnt /srv/www/distributions/<image_name>	186
4.10.7. umount /mnt	186
4.10.8. 특정 섹션 시작(주석임)	193

클라이언트 구성 안내서 개요

업데이트 날짜: 2023-04-18

Uyuni 설치 후 수행하는 첫 단계는 클라이언트를 등록하는 일입니다. Uyuni에서 등록 클라이언트를 관리하는 작업에 대부분의 시간을 소요합니다.

Uyuni는 다양한 클라이언트 기술과 호환됩니다. 즉, 다양한 하드웨어 옵션으로 SUSE Linux Enterprise 또는 다른 Linux 운영 체제를 실행하는 기존 또는 Salt 클라이언트를 설치할 수 있습니다.

지원되는 클라이언트 및 기능의 전체 목록은 [Client-configuration > Supported-features](#)를 참조하십시오.

이 안내서에서는 다양한 클라이언트를 수동 및 자동으로 등록하고 구성하는 방법에 대해 설명합니다.

Chapter 1. 지원되는 클라이언트 및 기능

Uyuni는 다양한 클라이언트 기술과 호환됩니다. 다양한 하드웨어 옵션으로 SUSE Linux Enterprise 또는 다른 Linux 운영 체제를 실행하는 기존 또는 Salt 클라이언트를 설치할 수 있습니다.

이 섹션에는 지원되는 클라이언트 시스템에 관한 요약이 포함되어 있습니다. 각 클라이언트에서 사용할 수 있는 기능의 상세 목록은 다음 페이지를 참조하십시오.

1.1. 지원되는 클라이언트 시스템

다음 테이블에는 기존 및 Salt 클라이언트에 지원되는 운영 체제가 나열됩니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 운영 체제를 실행하는 클라이언트를 SUSE에서 지원합니다.
- ✗ 이 운영 체제를 실행하는 클라이언트를 SUSE에서 지원하지 않습니다.
- ? 이 클라이언트는 현재 지원 고려 중이며, 이후에 지원될 수도 있고 지원되지 않을 수도 있습니다.



클라이언트 운영 체제 버전 및 SP 수준에는 Uyuni로 지원될 일반 지원(일반 또는 LTSS)이 적용되어야 합니다. 지원되는 제품 버전에 대한 상세 정보는 <https://www.suse.com/lifecycle>에서 참조하십시오.



클라이언트에서 실행되는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다.

표 1. 지원되는 클라이언트 시스템

Operating System	Architecture	Traditional Clients	Salt Clients
SUSE Linux Enterprise 15	x86-64, ppc64le, IBM Z, ARM	✓	✓
SUSE Linux Enterprise 12	x86-64, ppc64le, IBM Z, ARM	✓	✓
SUSE Linux Enterprise Server for SAP 15	x86-64, ppc64le	✓	✓
SUSE Linux Enterprise Server for SAP 12	x86-64, ppc64le	✓	✓
SLE Micro	x86-64, ppc64le, aarch64	✗	✓
openSUSE Leap 15	x86-64, aarch64	✓	✓
Alibaba Cloud Linux 2	x86-64, aarch64	✗	✓
AlmaLinux 9	x86-64, ppc64le, IBM Z, aarch64	✗	✓
AlmaLinux 8	x86-64, aarch64	✗	✓
Amazon Linux 2	x86-64, aarch64	✗	✓
CentOS 7	x86-64, ppc64le, aarch64	✓	✓
Debian 11	x86-64	✗	✓

Operating System	Architecture	Traditional Clients	Salt Clients
Debian 10	x86-64	✗	✓
Oracle Linux 9	x86-64, aarch64	✗	✓
Oracle Linux 8	x86-64, aarch64	✗	✓
Oracle Linux 7	x86-64, aarch64	✓	✓
Red Hat Enterprise Linux 9	x86-64	✗	✓
Red Hat Enterprise Linux 8	x86-64	✗	✓
Red Hat Enterprise Linux 7	x86-64	✓	✓
Rocky Linux 9	x86-64, aarch64, ppc64le, s390x	✗	✓
Rocky Linux 8	x86-64, aarch64	✗	✓
Ubuntu 22.04	amd64	✗	✓
Ubuntu 20.04	amd64	✗	✓
Ubuntu 18.04	amd64	✗	✓



Debian 및 Ubuntu에서는 x86-64 아키텍처가 amd64로 표시됩니다.

배포의 수명이 만료하면 3개월의 유예 기간이 시작되며 지원은 중단한 것으로 간주됩니다. 이 기간이 경과하면 해당 제품은 지원하지 않는 것으로 간주됩니다. 모든 지원은 최선의 노력으로 제공합니다.

수명 종료에 대한 자세한 설명은 <https://endoflife.software/operating-systems>를 참조하십시오.

1.2. 지원되는 도구 패키지

`spacewalk-utils` 및 `spacewalk-utils-extras` 패키지를 통해 추가 서비스 및 기능을 제공할 수 있습니다.

표 2. Spacewalk 유틸리티

도구 이름	설명	지원 여부
<code>spacewalk-common-channels</code>	SUSE Customer Center 가 제공하지 않는 채널 추가	✓
<code>spacewalk-hostname-rename</code>	Uyuni의 호스트 이름 변경	✓
<code>spacewalk-clone-by-date</code>	특정 날짜까지 채널 복제	✓
<code>spacewalk-sync-setup</code>	ISS 마스터 및 슬레이브 조직 매핑 설정	✓
<code>spacewalk-manage-channel-lifecycle</code>	채널 라이프사이클 관리	✓

1.3. 지원하는 SUSE 및 openSUSE 클라이언트 기능

아래 표는 SUSE 및 openSUSE 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다. 이 표에서는 SAP용 SLES, SLED, SUSE Linux Enterprise Server 및 HPC용 SUSE Linux Enterprise Server를 포함한 SUSE Linux Enterprise 운영 체제의 모든 변형에 대해 설명합니다.



클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. SUSE Linux Enterprise는 SUSE가 지원합니다. openSUSE는 SUSE 커뮤니티가 지원합니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 3. SUSE 및 openSUSE 운영 체제에서 지원하는 기능

기능	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15
클라이언트	✓	✓	✓
시스템 패키지	SUSE	SUSE	openSUSE Community
등록	✓	✓	Salt
패키지 설치	✓	✓	Salt
패치 적용	✓	✓	Salt
원격 명령	✓	✓	Salt
시스템 패키지 상태	Salt	Salt	Salt
시스템 사용자 정의 상태	Salt	Salt	Salt
그룹 사용자 정의 상태	Salt	Salt	Salt
조직 사용자 정의 상태	Salt	Salt	Salt
시스템 설정 관리자(SSM)	✓	✓	Salt
제품 마이그레이션	✓	✓	Salt
기본 가상 게스트 관리 *	✓	✓	Salt
고급 가상 게스트 관리 *	Salt	Salt	Salt
호스트 OS로 가상 게스트 설치(킥스타트)	Traditional	Traditional	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	Salt	Salt	Salt
가상 게스트 관리	Salt	Salt	Salt
시스템 배포(PXE/AutoYaST)	✓	✓	✓

기능	SUSE Linux Enterprise 12	SUSE Linux Enterprise 15	openSUSE 15
시스템 재배포(AutoYaST)	✓	✓	Salt
연락 방법	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	✓	✓	Salt
작업 체인	✓	✓	Salt
스테이징(패키지 사전 다운로드)	✓	✓	Salt
중복 패키지 보고	✓	✓	Salt
CVE 감사	✓	✓	Salt
SCAP 감사	✓	✓	Salt
패키지 검증	Traditional	Traditional	✗
패키지 잠금	Salt	Salt	Salt
시스템 잠금	Traditional	Traditional	✗
유지보수 기간	✓	✓	✓
시스템 스냅샷	Traditional	Traditional	✗
구성 파일 관리	✓	✓	Salt
패키지 프로필	Traditional. Salt: 프로필 지원됨, 동기화 지원되지 않음	Traditional. Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음
전원 관리	✓	✓	✓
모니터링 서버	Salt	Salt	Salt
모니터링되는 클라이언트	Salt	Salt	Salt
Docker 빌드호스트	Salt	Salt	?
OS를 사용한 Docker 이미지 빌드	Salt	Salt	Salt
Kiwi 빌드호스트	Salt	?	?
OS를 사용한 Kiwi 이미지 빌드	Salt	?	✗
반복 작업	Salt	Salt	Salt
AppStreams	N/A	N/A	N/A
Yomi	✗	✓	✓

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 설정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 설정이 포함됩니다.

1.4. Supported SLE Micro Client Features



- The operating system you run on a client is supported by the organization that supplies the operating system. SLE Micro is supported by SUSE.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 4. Supported Features on SLE Micro Operating Systems

Feature	SLE Micro
Client	Salt
Operating system packages	Salt
Registration	Salt
Install packages	Salt
Apply patches (requires CVE ID)	Salt
Remote commands	Salt
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	Salt
Product migration	Salt
Basic Virtual Guest Management *	Salt
Advanced Virtual Guest Management *	Salt
Virtual Guest Installation (Kickstart), as Host OS	Salt
Virtual Guest Installation (image template), as Host OS	Salt
System deployment (PXE/Kickstart)	Salt
System redeployment (Kickstart)	Salt

Feature	SLE Micro
Contact methods	Salt: ZeroMQ
Works with Uyuni Proxy	Salt
Action chains	Salt
Staging (pre-download of packages)	?
Duplicate package reporting	Salt
CVE auditing (requires CVE ID)	Salt
SCAP auditing	?
Package verification	?
Package locking	Salt
System locking	?
Maintenance Windows	?
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	Salt
Monitoring server	✗
Monitored clients	Salt
Docker buildhost	✗
Build Docker image with OS	✗
Kiwi buildhost	✗
Build Kiwi image with OS	Salt
Recurring Actions	Salt
AppStreams	N/A
Yomi	?

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 수정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 수정이 포함됩니다.

1.5. openSUSE Leap Micro Client Features



openSUSE Leap Micro is supported by the SUSE community.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 5. Supported Features on openSUSE Leap Micro Operating Systems

Feature	openSUSE Leap Micro
Client	Salt
Operating system packages	Salt
Registration	Salt
Install packages	Salt
Apply patches (requires CVE ID)	Salt
Remote commands	Salt
System package states	Salt
System custom states	Salt
Group custom states	Salt
Organization custom states	Salt
System set manager (SSM)	Salt
Product migration	Salt
Basic Virtual Guest Management *	Salt
Advanced Virtual Guest Management *	Salt
Virtual Guest Installation (Kickstart), as Host OS	Salt
Virtual Guest Installation (image template), as Host OS	Salt
System deployment (PXE/Kickstart)	Salt
System redeployment (Kickstart)	Salt
Contact methods	Salt: ZeroMQ
Works with Uyuni Proxy	Salt
Action chains	Salt
Staging (pre-download of packages)	?
Duplicate package reporting	Salt
CVE auditing (requires CVE ID)	Salt
SCAP auditing	?
Package verification	?

Feature	openSUSE Leap Micro
Package locking	Salt
System locking	?
Maintenance Windows	?
System snapshot	✗
Configuration file management	Salt
Snapshots and profiles	Salt: Profiles supported, Sync not supported
Power management	Salt
Monitoring server	✗
Monitored clients	Salt
Docker buildhost	✗
Build Docker image with OS	✗
Kiwi buildhost	✗
Build Kiwi image with OS	Salt
Recurring Actions	Salt
AppStreams	N/A
Yomi	?

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 수정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 수정이 포함됩니다.

1.6. 지원하는 Alibaba Cloud Linux 기능

아래 표는 Alibaba Cloud Linux 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다.



- 클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. Alibaba Cloud Linux는 Alibaba Cloud에서 지원됩니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.

- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 6. Alibaba Cloud Linux 운영 체제에서 지원하는 기능

기능	Alibaba Cloud Linux 2
클라이언트	Salt
운영 체제 패키지	Salt
등록	Salt
패키지 설치	Salt
패치 적용(CVE ID 필요)	Salt
원격 명령	Salt
시스템 패키지 상태	Salt
시스템 사용자 정의 상태	Salt
그룹 사용자 정의 상태	Salt
조직 사용자 정의 상태	Salt
시스템 설정 관리자(SSM)	Salt
제품 마이그레이션	N/A
기본 가상 게스트 관리 *	?
고급 가상 게스트 관리 *	?
호스트 OS로 가상 게스트 설치(킥스타트)	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	?
시스템 배포(PXE/킥스타트)	?
시스템 재배포(킥스타트)	?
연락 방법	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	Salt
작업 체인	Salt
스테이징(패키지 사전 다운로드)	Salt
중복 패키지 보고	Salt
CVE 감사(CVE ID 필요)	Salt
SCAP 감사	Salt
패키지 검증	✗
패키지 잠금	✗
시스템 잠금	✗
유지보수 기간	✓
시스템 스냅샷	✗
구성 파일 관리	Salt
스냅샷 및 프로필	Salt: 프로필 지원됨, 동기화 지원되지 않음

기능	Alibaba Cloud Linux 2
전원 관리	?
모니터링 서버	×
모니터링 클라이언트	Salt
Docker 빌드호스트	Salt
OS를 사용한 Docker 이미지 빌드	Salt
Kiwi 빌드호스트	Salt
OS를 사용한 Kiwi 이미지 빌드	Salt
반복 작업	Salt
AppStreams	N/A
Yomi	N/A

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 설정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 설정이 포함됩니다.

* 기존 스택은 Alibaba Cloud Linux에서 사용할 수 있지만 지원되지는 않습니다.

1.7. 지원하는 AlmaLinux 기능

아래 표는 AlmaLinux 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다.



클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다.
AlmaLinux는 AlmaLinux 커뮤니티에서 지원됩니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 7. AlmaLinux 운영 체제에서 지원하는 기능

기능	AlmaLinux 9	AlmaLinux 8
클라이언트	Salt (plain AlmaLinux)	Salt (plain AlmaLinux)

기능	AlmaLinux 9	AlmaLinux 8
시스템 패키지	AlmaLinux Community	AlmaLinux Community
등록	Salt	Salt
패키지 설치	Salt	Salt
패치 적용	Salt	Salt
원격 명령	Salt	Salt
시스템 패키지 상태	Salt	Salt
시스템 사용자 정의 상태	Salt	Salt
그룹 사용자 정의 상태	Salt	Salt
조직 사용자 정의 상태	Salt	Salt
시스템 설정 관리자(SSM)	Salt	Salt
제품 마이그레이션	N/A	N/A
기본 가상 게스트 관리 *	Salt	Salt
고급 가상 게스트 관리 *	Salt	Salt
호스트 OS로 가상 게스트 설치(킥스타트)	✗	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	Salt	Salt
시스템 배포(PXE/킥스타트)	Salt	Salt
시스템 재배포(킥스타트)	Salt	Salt
연락 방법	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	Salt	Salt
작업 체인	Salt	Salt
스테이징(패키지 사전 다운로드)	Salt	Salt
중복 패키지 보고	Salt	CVE 감사
Salt	Salt	SCAP 감사
Salt	Salt	패키지 검증
✗	✗	패키지 잠금
✗	✗	시스템 잠금
✗	✗	유지보수 기간
✓	✓	시스템 스냅샷
✗	✗	구성 파일 관리
Salt	Salt	스냅샷 및 프로필
Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음	전원 관리
Salt	Salt	모니터링 서버

기능	AlmaLinux 9	AlmaLinux 8
×	×	모니터링되는 클라이언트
Salt	Salt	Docker 빌드호스트
×	×	OS를 사용한 Kiwi 이미지 빌드
×	×	Kiwi 빌드호스트
×	×	OS를 사용한 Kiwi 이미지 빌드
×	×	반복 작업
Salt	Salt	AppStreams
✓	✓	Yomi

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 설정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 설정이 포함됩니다.

1.8. 지원되는 Amazon Linux 기능

아래 표는 Amazon Linux 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다.



- 클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. Amazon Linux은 Amazon에서 지원됩니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 8. Amazon Linux 운영 체제에서 지원하는 기능

기능	Amazon Linux 2
클라이언트	Salt
운영 체제 패키지	Salt
등록	Salt
패키지 설치	Salt

기능	Amazon Linux 2
패치 적용(CVE ID 필요)	Salt
원격 명령	Salt
시스템 패키지 상태	Salt
시스템 사용자 정의 상태	Salt
그룹 사용자 정의 상태	Salt
조직 사용자 정의 상태	Salt
시스템 설정 관리자(SSM)	Salt
제품 마이그레이션	N/A
기본 가상 게스트 관리 *	?
고급 가상 게스트 관리 *	?
호스트 OS로 가상 게스트 설치(킥스타트)	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	?
시스템 배포(PXE/킥스타트)	?
시스템 재배포(킥스타트)	?
연락 방법	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	Salt
작업 체인	Salt
스테이징(패키지 사전 다운로드)	Salt
중복 패키지 보고	Salt
CVE 감사(CVE ID 필요)	Salt
SCAP 감사	Salt
패키지 검증	✗
패키지 잠금	✗
시스템 잠금	✗
유지보수 기간	✓
시스템 스냅샷	✗
구성 파일 관리	Salt
스냅샷 및 프로필	Salt: 프로필 지원됨, 동기화 지원되지 않음
전원 관리	?
모니터링 서버	✗
모니터링되는 클라이언트	Salt
Docker 빌드호스트	Salt
OS를 사용한 Docker 이미지 빌드	Salt
Kiwi 빌드호스트	Salt

기능	Amazon Linux 2
OS를 사용한 Kiwi 이미지 빌드	Salt
반복 작업	Salt
AppStreams	N/A
Yomi	N/A

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프사이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 수정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프사이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 수정이 포함됩니다.

* 기존 스택은 Amazon Linux에서 사용할 수 있지만 지원되지는 않습니다.

1.9. 지원하는 CentOS 기능

아래 표는 CentOS 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다.



클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. CentOS는 CentOS 커뮤니티가 지원합니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 9. CentOS 운영 체제에서 지원하는 기능

기능	CentOS 7
클라이언트	✓ (plain CentOS)
시스템 패키지	CentOS Community
등록	✓
패키지 설치	✓
패치 적용(CVE ID 필요)	✓(오류를 위해 타사 서비스 필요)
원격 명령	✓
시스템 패키지 상태	Salt

기능	CentOS 7
시스템 사용자 정의 상태	Salt
그룹 사용자 정의 상태	Salt
조직 사용자 정의 상태	Salt
시스템 설정 관리자(SSM)	✓
제품 마이그레이션	N/A
기본 가상 게스트 관리 *	✓
고급 가상 게스트 관리 *	Salt
호스트 OS로 가상 게스트 설치(킥스타트)	Traditional
호스트 OS로 가상 게스트 설치(이미지 템플릿)	✓
시스템 배포(PXE/킥스타트)	✓
시스템 재배포(킥스타트)	✓
연락 방법	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	✓
작업 체인	✓
스테이징(패키지 사전 다운로드)	✓
중복 패키지 보고	✓
CVE 감사(CVE ID 필요)	✓
SCAP 감사	✓
패키지 검증	Traditional
패키지 잠금	✓
시스템 잠금	Traditional
유지보수 기간	✓
시스템 스냅샷	Traditional
구성 파일 관리	✓
스냅샷 및 프로필	Traditional. Salt: 프로필 지원됨, 동기화 지원되지 않음
전원 관리	✓
모니터링 서버	✗
모니터링되는 클라이언트	Salt
Docker 빌드호스트	✗
OS를 사용한 Docker 이미지 빌드	✗
Kiwi 빌드호스트	✗
OS를 사용한 Kiwi 이미지 빌드	✗
반복 작업	Salt

기능	CentOS 7
AppStreams	N/A
Yomi	N/A

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 수정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 수정이 포함됩니다.

1.10. 지원하는 Debian 기능

아래 표는 Debian 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다.



- 클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. Debian은 Debian 커뮤니티가 지원합니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 10. Debian 운영 체제에서 지원하는 기능

기능	Debian 10	Debian 11
클라이언트	✓	✓
시스템 패키지	Debian Community	Debian Community
등록	Salt	Salt
패키지 설치	Salt	Salt
패치 적용	?	?
원격 명령	Salt	Salt
시스템 패키지 상태	Salt	Salt
시스템 사용자 정의 상태	Salt	Salt
그룹 사용자 정의 상태	Salt	Salt
조직 사용자 정의 상태	Salt	Salt

기능	Debian 10	Debian 11
시스템 설정 관리자(SSM)	Salt	Salt
제품 마이그레이션	N/A	N/A
기본 가상 게스트 관리 *	Salt	Salt
고급 가상 게스트 관리 *	Salt	Salt
호스트 OS로 가상 게스트 설치(킥스타트)	✗	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	Salt	Salt
시스템 배포(PXE/킥스타트)	✗	✗
시스템 재배포(킥스타트)	✗	✗
연락 방법	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	Salt	Salt
작업 체인	Salt	Salt
스테이징(패키지 사전 다운로드)	Salt	Salt
중복 패키지 보고	Salt	Salt
CVE 감사	?	?
SCAP 감사	?	?
패키지 검증	✗	✗
패키지 잠금	✓	✓
시스템 잠금	✗	✗
유지보수 기간	✓	✓
시스템 스냅샷	✗	✗
구성 파일 관리	Salt	Salt
패키지 프로필	Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음
전원 관리	✓	✓
모니터링 서버	✗	✗
모니터링되는 클라이언트	Salt	Salt
Docker 빌드호스트	?	?
OS를 사용한 Docker 이미지 빌드	Salt	Salt
Kiwi 빌드호스트	✗	✗
OS를 사용한 Kiwi 이미지 빌드	✗	✗
반복 작업	Salt	Salt
AppStreams	N/A	N/A
Yomi	N/A	N/A

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 수정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 수정이 포함됩니다.

1.11. 지원하는 Oracle 기능

아래 표는 Oracle Linux 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다.



- 클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. Oracle Linux는 Oracle이 지원합니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 11. Oracle Linux 운영 체제에서 지원하는 기능

기능	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
클라이언트	✓	Salt	Salt
운영 체제 패키지	✓	Salt	Salt
등록	✓	Salt	Salt
패키지 설치	✓	Salt	Salt
패치 적용(CVE ID 필요)	✓	Salt	Salt
원격 명령	✓	Salt	Salt
시스템 패키지 상태	Salt	Salt	Salt
시스템 사용자 정의 상태	Salt	Salt	Salt
그룹 사용자 정의 상태	Salt	Salt	Salt
조직 사용자 정의 상태	Salt	Salt	Salt
시스템 설정 관리자(SSM)	✓	Salt	Salt
제품 마이그레이션	N/A	N/A	N/A
기본 가상 게스트 관리 *	✓	Salt	Salt
고급 가상 게스트 관리 *	Salt	Salt	Salt

기능	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
호스트 OS로 가상 게스트 설치(킥스타트)	Traditional	✗	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	✓	Salt	Salt
시스템 배포(PXE/킥스타트)	✓	Salt	Salt
시스템 재배포(킥스타트)	✓	Salt	Salt
연락 방법	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	✓	Salt	Salt
작업 체인	✓	Salt	Salt
스테이징(패키지 사전 다운로드)	✓	Salt	Salt
중복 패키지 보고	✓	Salt	Salt
CVE 감사(CVE ID 필요)	✓	Salt	Salt
SCAP 감사	✓	Salt	Salt
패키지 검증	Traditional	✗	✗
패키지 잠금	✓	?	?
시스템 잠금	Traditional	✗	✗
유지보수 기간	✓	✓	✓
시스템 스냅샷	Traditional	✗	✗
구성 파일 관리	✓	Salt	Salt
스냅샷 및 프로필	Traditional. Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음
전원 관리	✓	Salt	Salt
모니터링 서버	✗	✗	✗
모니터링되는 클라이언트	Salt	Salt	Salt
Docker 빌드호스트	✗	✗	✗
OS를 사용한 Docker 이미지 빌드	✗	✗	✗
Kiwi 빌드호스트	✗	✗	✗
OS를 사용한 Kiwi 이미지 빌드	✗	✗	✗
반복 작업	Salt	Salt	Salt
AppStreams	N/A	✓	✓

기능	Oracle Linux 7	Oracle Linux 8	Oracle Linux 9
Yomi	N/A	N/A	N/A

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 설정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 설정이 포함됩니다.

1.12. 지원하는 Red Hat Enterprise Linux 기능

This table lists the availability of various features on native Red Hat Enterprise Linux clients.



- 클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. Red Hat Enterprise Linux는 Red Hat이 지원합니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 12. Red Hat Enterprise Linux 운영 체제에서 지원하는 기능

기능	RHEL 7	RHEL 8	RHEL 9
클라이언트	✓	Salt	Salt
시스템 패키지	Red Hat	Red Hat	Red Hat
등록	✓	Salt	Salt
패키지 설치	✓	Salt	Salt
패치 적용	✓	Salt	Salt
원격 명령	✓	Salt	Salt
시스템 패키지 상태	Salt	Salt	Salt
시스템 사용자 정의 상태	Salt	Salt	Salt
그룹 사용자 정의 상태	Salt	Salt	Salt
조직 사용자 정의 상태	Salt	Salt	Salt
시스템 설정 관리자(SSM)	Salt	Salt	Salt

기능	RHEL 7	RHEL 8	RHEL 9
제품 마이그레이션	N/A	N/A	N/A
기본 가상 게스트 관리 *	✓	Salt	Salt
고급 가상 게스트 관리 *	Salt	Salt	Salt
호스트 OS로 가상 게스트 설치(킥스타트)	Traditional	✗	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	✓	Salt	Salt
시스템 배포(PXE/킥스타트)	✓	Salt	Salt
시스템 재배포(킥스타트)	✓	Salt	Salt
연락 방법	Traditional: OSAD, RHNSD, SSH-push. Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	✓	Salt	Salt
작업 체인	✓	Salt	Salt
스테이징(패키지 사전 다운로드)	✓	Salt	Salt
중복 패키지 보고	✓	Salt	Salt
CVE 감사	✓	Salt	Salt
SCAP 감사	✓	Salt	Salt
패키지 검증	Traditional	✗	✗
패키지 잠금	✓	?	?
시스템 잠금	Traditional	✗	✗
유지보수 기간	✓	✓	✓
시스템 스냅샷	Traditional	✗	✗
구성 파일 관리	✓	Salt	Salt
스냅샷 및 프로필	Traditional. Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음
전원 관리	✓	Salt	Salt
모니터링 서버	✗	✗	✗
모니터링되는 클라이언트	Salt	Salt	Salt
Docker 빌드호스트	✗	✗	✗
OS를 사용한 Docker 이미지 빌드	?	?	?
Kiwi 빌드호스트	✗	✗	✗

기능	RHEL 7	RHEL 8	RHEL 9
OS를 사용한 Kiwi 이미지 빌드	✗	✗	✗
반복 작업	Salt	Salt	Salt
AppStreams	N/A	✓	✓
Yomi	N/A	N/A	N/A

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 설정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 설정이 포함됩니다.

1.13. 지원하는 Rocky Linux 기능

아래 표는 Rocky Linux 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다.



- 클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. Rocky Linux는 Rocky Linux 커뮤니티가 지원합니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 13. Rocky Linux 운영 체제에서 지원하는 기능

기능	Rocky Linux 8	Rocky Linux 9
클라이언트	Salt (plain Rocky Linux)	Salt (plain Rocky Linux)
시스템 패키지	Rocky Linux Community	Rocky Linux Community
등록	Salt	Salt
패키지 설치	Salt	Salt
패치 적용	Salt	Salt
원격 명령	Salt	Salt
시스템 패키지 상태	Salt	Salt
시스템 사용자 정의 상태	Salt	Salt

기능	Rocky Linux 8	Rocky Linux 9
그룹 사용자 정의 상태	Salt	Salt
조직 사용자 정의 상태	Salt	Salt
시스템 설정 관리자(SSM)	Salt	Salt
제품 마이그레이션	N/A	N/A
기본 가상 게스트 관리 *	Salt	Salt
고급 가상 게스트 관리 *	Salt	Salt
호스트 OS로 가상 게스트 설치(킥스타트)	✗	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	Salt	Salt
시스템 배포(PXE/킥스타트)	Salt	Salt
시스템 재배포(킥스타트)	Salt	Salt
연락 방법	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	Salt	Salt
작업 체인	Salt	Salt
스테이징(패키지 사전 다운로드)	Salt	Salt
중복 패키지 보고	Salt	Salt
CVE 감사	Salt	Salt
SCAP 감사	Salt	Salt
패키지 검증	✗	✗
패키지 잠금	?	?
시스템 잠금	✗	✗
유지보수 기간	✓	✓
시스템 스냅샷	✗	✗
구성 파일 관리	Salt	Salt
스냅샷 및 프로필	Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음
전원 관리	Salt	Salt
모니터링 서버	✗	✗
모니터링되는 클라이언트	Salt	Salt
Docker 빌드호스트	✗	✗
OS를 사용한 Docker 이미지 빌드	✗	✗
Kiwi 빌드호스트	✗	✗
OS를 사용한 Kiwi 이미지 빌드	✗	✗
반복 작업	Salt	Salt

기능	Rocky Linux 8	Rocky Linux 9
AppStreams	✓	✓
Yomi	N/A	N/A

* 가상 게스트 관리:

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 수정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 수정이 포함됩니다.

1.14. 지원하는 Ubuntu 기능

아래 표는 Ubuntu 클라이언트에서 다양한 기능의 사용 가능 여부를 정리한 것입니다.



클라이언트에서 실행하는 운영 체제는 운영 체제를 제공하는 조직이 지원합니다. Ubuntu는 Canonical이 지원합니다.

이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- ✓ 이 기능을 Salt 및 기존 클라이언트 모두에서 사용할 수 있습니다.
- ✗ 이 기능을 사용할 수 없습니다.
- ? 이 기능은 현재 고려 중이며, 이후에 사용할 수도 있고 사용하지 못할 수도 있습니다.
- Traditional 이 기능은 기존 클라이언트에서만 지원합니다.
- Salt 이 기능은 Salt 클라이언트에서만 지원합니다.

표 14. Ubuntu 운영 체제에서 지원하는 기능

기능	Ubuntu 18.04	Ubuntu 20.04	Ubuntu 22.04
클라이언트	✓	✓	✓
시스템 패키지	Canonical	Canonical	Canonical
등록	Salt	Salt	Salt
패키지 설치	Salt	Salt	Salt
패치 적용	✓	✓	✓
원격 명령	Salt	Salt	Salt
시스템 패키지 상태	Salt	Salt	Salt
시스템 사용자 정의 상태	Salt	Salt	Salt
그룹 사용자 정의 상태	Salt	Salt	Salt
조직 사용자 정의 상태	Salt	Salt	Salt

기능	Ubuntu 18.04	Ubuntu 20.04	Ubuntu 22.04
시스템 설정 관리자(SSM)	Salt	Salt	Salt
제품 마이그레이션	N/A	N/A	N/A
기본 가상 게스트 관리 *	Salt	Salt	Salt
고급 가상 게스트 관리 *	Salt	Salt	Salt
호스트 OS로 가상 게스트 설치(킥스타트)	✗	✗	✗
호스트 OS로 가상 게스트 설치(이미지 템플릿)	Salt	Salt	Salt
시스템 배포(PXE/킥스타트)	✗	✗	✗
시스템 재배포(킥스타트)	✗	✗	✗
연락 방법	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH	Salt: ZeroMQ, Salt-SSH
Uyuni Proxy와 함께 작동	Salt	Salt	Salt
작업 체인	Salt	Salt	Salt
스테이징(패키지 사전 다운로드)	Salt	Salt	Salt
중복 패키지 보고	Salt	Salt	Salt
CVE 감사	?	?	?
SCAP 감사	?	?	?
패키지 검증	✗	✗	✗
패키지 잠금	✓	✓	✓
시스템 잠금	✗	✗	✗
시스템 스냅샷	✗	✗	✗
구성 파일 관리	Salt	Salt	Salt
패키지 프로필	Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음	Salt: 프로필 지원됨, 동기화 지원되지 않음
전원 관리	✓	✓	✓
모니터링	Salt	Salt	Salt
Docker 빌드호스트	?	?	?
OS를 사용한 Docker 이미지 빌드	Salt	Salt	Salt
Kiwi 빌드호스트	✗	✗	✗
OS를 사용한 Kiwi 이미지 빌드	✗	✗	✗
반복 작업	Salt	Salt	Salt
AppStreams	N/A	N/A	N/A
Yomi	N/A	N/A	N/A

*** 가상 게스트 관리:**

이 표에서 가상 게스트 관리는 기본과 고급으로 구분되어 있습니다.

기본 가상 게스트 관리에는 VM, 느린 새로 고침, VM 라이프싸이클 작업(시작, 중지, 다시 시작, 일시 중지) 나열과 VM vCPU 및 메모리 수정이 포함됩니다.

고급 가상 게스트 관리에는 기본 가상 게스트 관리의 모든 기능과 함께 빠른 새로 고침, VM 라이프싸이클 작업(삭제, 재설정, 전원 끄기), VM 디스크, 네트워크, 그래픽 표시 및 그래픽 표시 구성 수정이 포함됩니다.

Chapter 2. 구성 기본

Uyuni에서는 클라이언트 등록을 위한 환경을 준비하기 위해 여러 단계를 거친 후 매우 다양한 작업을 활용해야 합니다.

이 섹션에는 성공적인 Uyuni 설치 및 설정 후 환경 작업을 지원하기 위해 필요한 초기 구성 단계에 대한 요약이 포함되어 있습니다.

- Uyuni 설치에 대한 자세한 정보는 [Installation-and-upgrade > Install-uyuni](#)에서 참조하십시오.
- Uyuni 설정에 대한 자세한 정보는 [Installation-and-upgrade > Uyuni-server-setup](#)에서 참조하십시오.

2.1. 소프트웨어 채널

채널은 소프트웨어 패키지를 그룹화하는 방법입니다. 소프트웨어 패키지는 리포지토리가 제공하며, 리포지토리는 채널과 연결되어 있습니다. 클라이언트를 소프트웨어 채널에 가입하면 클라이언트는 자신과 연결된 모든 소프트웨어를 설치하고 업데이트할 수 있습니다.

Uyuni에서 채널은 기본 채널과 하위 채널로 구분됩니다. 이런 식으로 채널을 구성하면 호환되는 패키지만 각 시스템에 설치됩니다. 클라이언트는 등록 중에 클라이언트 운영 체제 및 아키텍처에 따라 할당된 하나의 기본 채널에만 가입되어야 합니다. 벤더가 제공하는 유료 채널에 대해서는 연결된 구독이 있어야 합니다.

기본 채널은 특정 운영 체제 유형, 버전 및 아키텍처를 위해 구축된 패키지로 구성됩니다. 예를 들어 SUSE Linux Enterprise Server 15 x86-64 기본 채널에는 이 운영 체제 및 아키텍처와 호환되는 소프트웨어만 포함되어 있습니다.

하위 채널은 기본 채널과 연결되어 있으며 기본 채널과 호환되는 패키지만 제공합니다. 시스템은 기본 채널의 여러 하위 채널에 가입할 수 있습니다. 시스템이 기본 채널에 할당된 경우 해당 시스템은 관련 하위 채널만 설치할 수 있습니다. 예를 들어 시스템에 SUSE Linux Enterprise Server 15 **x86_64** 기본 채널이 할당된 경우 호환되는 기본 채널이나 기본 채널과 연결된 하위 채널 중 하나를 통해 사용할 수 있는 패키지만 설치하거나 업데이트할 수 있습니다.

Uyuni Web UI에서 **소프트웨어** > **채널 목록** > **전체**로 이동하여 사용 가능한 채널을 검색할 수 있습니다. **소프트웨어** > **관리** > **채널**로 이동하여 새 채널을 수정하거나 생성할 수 있습니다.

사용자 지정 채널을 포함한 채널 사용에 대한 자세한 내용은 [Administration > Channel-management](#)을 참조하십시오.

2.1.1. SUSE Package Hub가 제공하는 패키지

SUSE Package Hub는 SUSE Linux Enterprise 제품의 확장으로, openSUSE 커뮤니티가 제공하는 추가 오픈 소스 소프트웨어를 제공합니다.



- SUSE Package Hub의 패키지는 openSUSE 커뮤니티가 제공합니다. 이 패키지는 SUSE에서 지원되지 않습니다.

클라이언트에서 SUSE Linux Enterprise 운영 체제를 사용 중인 경우 SUSE Package Hub 확장을 활성화하여 이러한 추가 패키지에 액세스할 수 있습니다. 이렇게 하면 클라이언트를 가입할 수 있는 SUSE Package Hub 채널이 제공됩니다.

SUSE Package Hub는 다수의 패키지를 제공하므로 동기화하는 데 시간이 오래 걸리고 대량의 디스크 공간을 소비할 수 있습니다. SUSE Package Hub가 제공하는 패키지가 필요하지 않은 경우 SUSE Package Hub를 활성화하지 마십시오.

지원되지 않는 패키지를 의도치 않게 설치하거나 업데이트하지 않도록 하려면 처음에 모든 SUSE Package Hub 패키지를 거부하는 컨텐트 라이프사이클 관리 전략을 구현하는 것이 좋습니다. 그러면 필요한 특정 패키지를 명시적으로 활성화할 수 있습니다. 컨텐트 라이프사이클 관리에 대한 자세한 내용은 **Administration > Content-lifecycle**을 참조하십시오.

2.1.2. AppStream이 제공하는 패키지

Red Hat 기반 클라이언트의 경우 AppStream을 통해 추가 패키지가 제공됩니다. 대부분의 경우 AppStream 패키지가 있어야 필요한 소프트웨어를 모두 보유할 수 있습니다.

Uyuni Web UI에서 AppStream 패키지를 관리할 때 패키지 업데이트에 대한 모순되는 제안을 볼 수도 있습니다. 이는 Uyuni(가) 모듈 메타데이터를 올바르게 해석할 수 없기 때문입니다. 컨텐트 라이프사이클 관리(CLM) AppStream 필터를 사용하여 AppStream 리포지토리를 일부 업그레이드 작업에서 사용할 비모듈형 리포지토리로 변환할 수 있습니다. CLM AppStream 필터에 대한 자세한 설명은 **Administration > Content-lifecycle-examples**에서 참조하십시오.

2.1.3. EPEL이 제공하는 패키지

Red Hat 기반 클라이언트의 경우 EPEL(엔터프라이즈 Linux용 추가 패키지)을 통해 추가 패키지가 제공됩니다. EPEL은 추가 소프트웨어를 제공하는 옵션 패키지 리포지토리입니다.



- EPEL의 패키지는 Fedora 커뮤니티가 제공합니다. 이 패키지는 SUSE에서 지원하지 않습니다.

클라이언트에서 Red Hat 운영 체제를 사용 중인 경우 EPEL 확장을 활성화하여 이러한 추가 패키지에 액세스할 수 있습니다. 이렇게 하면 클라이언트를 가입할 수 있는 EPEL 채널이 제공됩니다.

EPEL은 다수의 패키지를 제공하므로 동기화하는 데 시간이 오래 걸리고 대량의 디스크 공간을 소비할 수 있습니다. EPEL이 제공하는 패키지가 필요하지 않은 경우 EPEL 리포지토리를 활성화하지 마십시오.

지원되지 않는 패키지를 의도치 않게 설치하거나 업데이트하지 않도록 하려면 처음에 모든 EPEL 패키지를 거부하는 컨텐트 라이프사이클 관리(CLM) 전략을 구현하는 것이 좋습니다. 그러면 필요한 특정 패키지를 명시적으로 활성화할 수 있습니다. 컨텐트 라이프사이클 관리에 대한 자세한 내용은 **Administration > Content-lifecycle**을 참조하십시오.

2.1.4. SUSE Linux Enterprise 클라이언트의 Unified Installer 업데이트 채널

이 채널은 운영 체제를 설치하기 전에 최신 상태인지 확인하기 위해 Unified Installer에서 사용합니다. 모든 SUSE Linux Enterprise 제품은 설치 중에 설치 프로그램 업데이트 채널에 액세스할 수 있는 권한이 있어야 합니다.

SUSE Linux Enterprise Server 클라이언트의 경우 설치 프로그램 업데이트 채널은 이 채널을 포함하는 제품을 추가하면 기본적으로 동기화되고 이 제품 채널로 자동 설치 가능한 배포판을 생성하면 활성화됩니다.

SAP용 SUSE Linux Enterprise를 포함한 다른 모든 SUSE Linux Enterprise 변형의 경우 설치 프로그램 업데이트 채널을 수동으로 추가해야 합니다. 이 작업을 수행하려면 SUSE Linux Enterprise 변형의 기본 채널 아래에 적절한 SUSE Linux Enterprise Server 설치 프로그램 업데이트 채널을 복제해야 합니다. 채널을 복제한 후 SUSE Linux Enterprise 변형에 대해 자동 설치 가능한 배포판을 생성할 때 이 채널이 자동으로 사용됩니다.

2.1.5. 소프트웨어 리포지토리

리포지토리는 소프트웨어 패키지를 수집하는 데 사용됩니다. 소프트웨어 리포지토리에 액세스할 수 있으면 리포지토리가 제공하는 모든 소프트웨어를 설치할 수 있습니다. 클라이언트를 채널에 할당하고 클라이언트에서 패키지를 설치하고 업데이트하려면 Uyuni에 소프트웨어 채널과 연결된 리포지토리가 최소 한 개 있어야 합니다.

Uyuni의 기본 채널은 대부분 이미 올바른 리포지토리에 연결되어 있습니다. 사용자 정의 채널을 생성하려면 액세스할 수 있는 또는 사용자가 직접 생성한 리포지토리를 연결해야 합니다.

사용자 정의 리포지토리에 대한 자세한 내용은 [Administration > Custom-channels](#)에서 참조하십시오.

2.1.5.1. 로컬 리포지토리 위치

Salt 클라이언트에서 로컬 리포지토리를 구성하여 Uyuni 채널이 공급하지 않는 패키지를 제공할 수 있습니다.



대부분의 경우 클라이언트 시스템에는 로컬 리포지토리가 필요 없습니다. 로컬 리포지토리로 인해 클라이언트에서 어떤 패키지를 사용할 수 있는지 파악하는데 문제가 생길 수 있습니다. 그러면 결국 예상치 않은 패키지를 설치하게 될 수 있습니다.

온보딩 중에는 로컬 리포지토리가 비활성화됩니다.

Salt 클라이언트의 경우 채널 상태가 실행될 때마다 로컬 리포지토리가 비활성화됩니다. 예를 들어, highstate를 적용하거나 패키지 작업을 수행하는 경우가 이에 해당합니다.

온보딩 후 로컬 리포지토리를 활성화 상태로 유지해야 하는 경우 영향을 받는 Salt 클라이언트에 대해 설정해야 하는 열은 다음과 같습니다.

`/srv/pillar/top.sls` 파일 편집:

```
base:
  'minionid':
    - localrepos
```

`/srv/pillar/localrepos.sls` 파일 편집:

```
mgr_disable_local_repos: False
```

클라이언트가 온보딩을 완료하고 나면 다음 위치에 로컬 리포지토리를 추가할 수 있습니다.

표 15. 로컬 리포지토리 위치

Client Operating System	Local Repository Directory
SUSE Linux Enterprise Server	<code>/etc/zypp/repos.d</code>
openSUSE	<code>/etc/zypp/repos.d</code>
Red Hat Enterprise Linux	<code>/etc/yum.repos.d/</code>
CentOS	<code>/etc/yum.repos.d/</code>
Ubuntu	<code>/etc/apt/sources.list.d/</code>

Client Operating System	Local Repository Directory
Debian	/etc/apt/sources.list.d/

2.1.6. 소프트웨어 제품

Uyuni에서는 소프트웨어를 제품 내에서 제공합니다. SUSE 구독을 통해 다양한 제품에 액세스할 수 있습니다. 이러한 제품은 Uyuni Web UI에서 **관리 > 설치 마법사 > 제품**으로 이동하여 검색하고 선택할 수 있습니다.

제품에는 소프트웨어 채널이 포함되어 있는데, 그 개수는 다양합니다. **제품 채널 표시** 아이콘을 클릭하여 제품에 포함된 채널을 확인합니다. 제품을 추가하고 동기화를 완료하고 나면 제품이 제공하는 채널에 액세스할 수 있고 Uyuni 서버 및 클라이언트에서 제품에 있는 패키지를 사용할 수 있습니다.

절차: 소프트웨어 채널 추가

1. Uyuni Web UI에서 **관리 > 설치 마법사 > 제품**으로 이동합니다.
2. 검색 창을 사용하여 클라이언트 운영 체제 및 아키텍처에 적합한 제품을 찾고 해당 제품을 확인하십시오. 모든 필수 채널은 자동으로 확인됩니다. 또한, **추천 포함** 토글이 켜져 있으면 모든 추천 채널이 확인됩니다. 관련 제품의 전체 목록을 보려면 화살표를 클릭하고 필요한 추가 제품이 선택되어 있는지 확인하십시오.
3. **제품 추가**를 클릭하고 제품이 동기화를 마칠 때까지 기다립니다.

자세한 설명은 **Installation-and-upgrade > Setup-wizard**를 참조하십시오.

2.2. 부트스트랩 리포지토리

부트스트랩 리포지토리에는 부트스트랩 중에 Salt 또는 기존 클라이언트를 등록하기 위한 필수 패키지와 클라이언트에 Salt를 설치하기 위한 패키지가 포함되어 있습니다. 제품이 동기화되면 부트스트랩 리포지토리가 자동으로 생성되고 Uyuni 서버에서 다시 생성됩니다.

2.2.1. 부트스트랩 리포지토리 생성 준비

동기화를 위해 제품을 선택하면 모든 필수 채널이 완전히 미러링되자마자 부트스트랩 리포지토리가 자동으로 생성됩니다.

절차: Web UI에서 동기화 진행률 확인

1. Uyuni Web UI에서 **소프트웨어 > 관리 > 채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
2. **리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

2.2.2. 자동 모드를 위한 옵션

자동 부트스트랩 리포지토리가 생성되는 방식을 변경할 수 있습니다. 이 섹션에서는 다양한 설정에 대해 자세히 설명합니다.

플러시 모드::

플러시 모드

기본적으로 기존 리포지토리는 최신 패키지로만 업데이트됩니다. 대신, 빈 리포지토리로 항상 시작하도록 구성할 수 있습니다. 이 동작을 활성화하려면 다음과 같이 `/etc/rhn/rhn.conf`에서 이 값을 추가하거나 편집하십시오.

```
server.susemanager.bootstrap_repo_flush = 1
```

자동 모드::

자동 모드

기본적으로 부트스트랩 리포지토리 자동 재생성은 활성화되어 있습니다. 이 작업을 비활성화하려면 다음과 같이 `/etc/rhn/rhn.conf`에서 이 값을 추가하거나 편집하십시오.

```
server.susemanager.auto_generate_bootstrap_repo = 0
```

2.2.2.1. 부트스트랩 데이터 파일 구성

도구는 각 배포에 어떤 패키지가 필요한지에 대한 정보가 포함된 데이터 파일을 사용합니다. 이 데이터 파일은 `/usr/share/susemanager/mgr_bootstrap_data.py`에 저장되어 있습니다. SUSE는 이 파일을 정기적으로 업데이트합니다. 이 파일을 변경하고 싶은 경우 직접 편집하지 마십시오. 대신, 다음과 같이 동일한 디렉토리에 사본을 생성하고 이 사본을 편집하십시오.

```
cd /usr/share/susemanager/
cp mgr_bootstrap_data.py my_data.py
```

변경을 완료했으면 새 파일을 사용하도록 Uyuni(를) 구성하십시오. 다음과 같이 `/etc/rhn/rhn.conf`에서 이 값을 추가하거나 편집하십시오.

```
server.susemanager.bootstrap_repo_datamodule = my_data
```



- 다음 업데이트에서 SUSE의 새 데이터는 새 데이터 파일이 아닌 원본 데이터 파일을 덮어씁니다. SUSE에서 제공하는 변경 사항으로 새 파일을 최신 상태로 유지해야 합니다.

2.2.3. 부트스트랩 리포지토리를 수동으로 생성

기본적으로 부트스트랩 리포지토리는 매일 다시 생성됩니다. 명령 프롬프트에서 부트스트랩 리포지토리를 수동으로 생성할 수 있습니다.

절차: SUSE Linux Enterprise용 부트스트랩 리포지토리 생성

- Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 부트스트랩 리포지토리를 생성하는 데 사용 가능한 배포를 나열합니다.

```
mgr-create-bootstrap-repo -l
```

- 다음과 같이 적절한 리포지토리 이름을 제품 레이블로 사용해 부트스트랩 리포지토리를 생성합니다.

```
mgr-create-bootstrap-repo -c SLE-version-x86_64
```

- 또는 사용 가능한 배포 목록에서 배포 이름 옆에 표시된 번호를 사용하십시오.

클라이언트 리포지토리는 </srv/www/htdocs/pub/repositories/>에 있습니다.

제품을 두 개 이상 미러링한 경우(예: SLES, SAP용 SLES) 또는 사용자 지정 채널을 사용하는 경우 부트스트랩 리포지토리를 생성할 때 사용할 상위 채널을 지정해야 합니다. 이는 모든 상황에서 필수가 아닙니다. 예를 들어, 일부 SLES 15 버전에는 공통 코드 기반이 있으므로, 상위 채널을 지정할 필요가 없습니다. 환경에서 필요한 경우에만 이 절차를 사용하십시오.

선택 사항 절차: 부트스트랩 리포지토리에 상위 채널 지정

- 사용 가능한 상위 채널을 확인하십시오.

```
mgr-create-bootstrap-repo -c SLE-15-x86_64
```

여러 개의 상위 채널용 옵션을 찾았습니다.

--with-parent-channel <label>옵션을 사용하고 다음 중 하나를 선택하십시오.

- sle-product-sles15-pool-x86_64
- sle-product-sles_sap15-pool-x86_64
- sle-product-sled15-pool-x86_64

- 다음과 같이 적절한 상위 채널을 지정합니다.

```
mgr-create-bootstrap-repo -c SLE-15-x86_64 --with-parent-channel sle-product-sled15-pool-x86_64
```

2.2.3.1. 아키텍처가 여러 개인 리포지토리

아키텍처가 여러 개 포함된 부트스트랩 리포지토리를 생성하는 경우 모든 아키텍처가 올바르게 업데이트되는지 주의 깊게 살펴봐야 합니다. 예를 들어 SLE용 IBM Z 및 x86-64 아키텍처가 </srv/www/htdocs/pub/repositories/sle/15/2/bootstrap/>에서 동일한 부트스트랩 리포지토리 URL을 사용합니다.

flush 옵션이 활성화된 상태에서 여러 아키텍처에 대해 부트스트랩 리포지토리를 생성하려고 하면 하나의 아키텍처만 생성됩니다. 이를 방지하려면 추가 아키텍처를 생성할 때 명령 프롬프트에서 **--no-flush** 옵션을 사용하십시오. 예:

```
mgr-create-bootstrap-repo -c SLE-15-SP2-x86_64
mgr-create-bootstrap-repo --no-flush -c SLE-15-SP2-s390x
```

2.2.4. 부트스트랩 및 사용자 지정 채널

사용자 지정 채널을 사용 중인 경우 **--with-custom-channels** 옵션을 **mgr-create-bootstrap-repo** 명령과 함께 사용할 수 있습니다. 이 경우 사용할 상위 채널도 지정해야 합니다.

사용자 지정 채널을 사용 중인 경우 부트스트랩 리포지토리 자동 생성에 실패할 수 있습니다. 이러한 경우에는 리포지토리를 수동으로 생성해야 합니다.

사용자 지정 채널에 대한 자세한 내용은 **Administration > Custom-channels**에서 참조하십시오.

2.3. 활성화 키

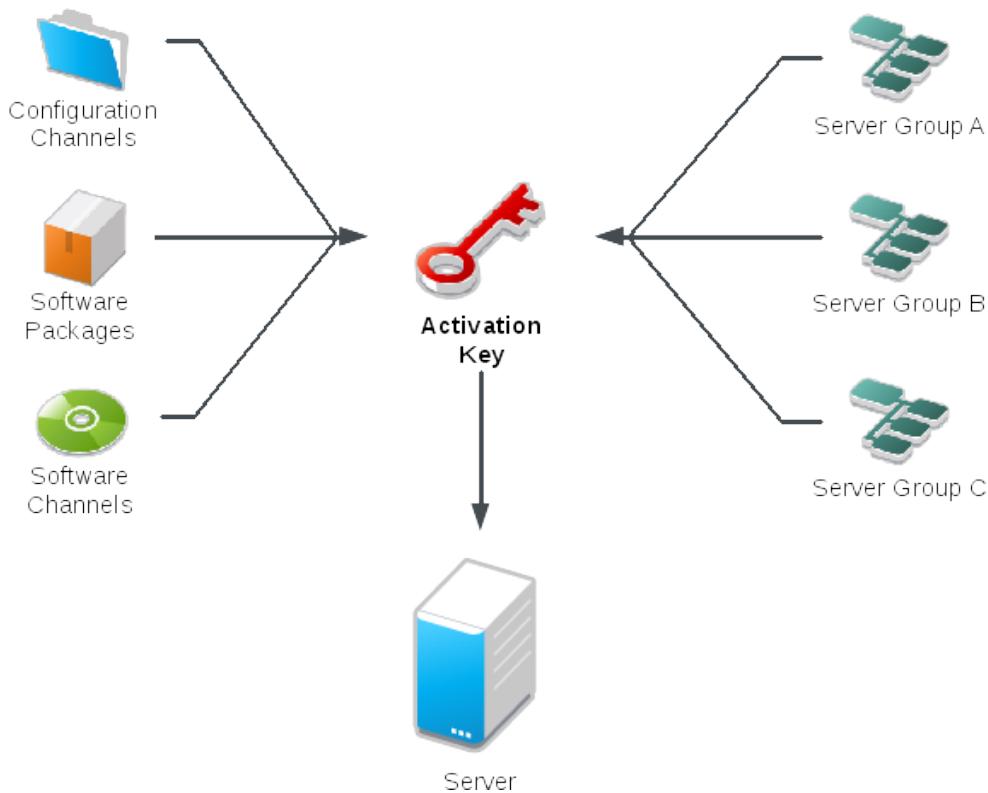
활성화 키는 기존 및 Salt 클라이언트에서 클라이언트가 올바른 소프트웨어 자격이 있는지, 적절한 채널에 연결하고 있는지, 관련 그룹에 가입되어 있는지 확인하는 데 사용됩니다. 각 활성화 키는 조직에 구속되며 키를 생성할 때 설정할 수 있습니다.

Uyuni에서 활성화 키는 레이블이 있는 구성 설정 그룹입니다. 레이블을 부트스트랩 스크립트에 파라미터로 추가하여 활성화 키와 연결된 모든 구성 설정을 적용할 수 있습니다. 활성화 키 레이블을 부트스트랩 스크립트와 함께 사용하는 것이 좋습니다. 부트스트랩 스크립트가 실행될 때 레이블에 연결된 모든 구성 설정은 스크립트가 실행되는 시스템에 적용됩니다.

활성화 키는 다음을 지정할 수 있습니다.

- 채널 할당
- 시스템 유형 또는 추가 자격
- 연락 방법
- 구성 파일
- 설치할 패키지
- 시스템 그룹 할당

활성화 키는 클라이언트 등록 시 사용되며 다시 사용되지 않습니다. 클라이언트는 등록된 후에 지정된 활성화 키와 관계없이 변경할 수 있습니다. 활성화 키와 클라이언트 사이의 관계는 기록 목적으로만 기록됩니다.



절차: 활성화 키 생성

1. Uyuni Web UI에서 관리자 권한으로 **시스템 > 활성화 키**로 이동합니다.
2. **키 생성** 버튼을 클릭합니다.
3. **활성화 키 상세 정보** 페이지의 **설명** 필드에 활성화 키의 설명을 입력합니다.
4. **키** 필드에 활성화 키의 이름을 입력합니다. 예를 들어, SUSE Linux Enterprise Server 15 SP4에 대해서는 **SLES15-SP4**를 입력합니다.



SUSE 제품인 경우 **키** 필드에 쉼표를 사용하지 마십시오. 하지만 Red Hat 제품인 경우 **반드시** 쉼표를 사용해야 합니다.

- 기타 모든 문자는 허용되지만 **<> (){}<** (공백 포함)는 자동으로 제거됩니다.
- 필드가 비어 있으면 임의의 스트링이 생성됩니다.

5. **기본 채널** 드롭다운 상자에서 적절한 기본 소프트웨어 채널을 선택하고, 관련 하위 채널이 작성되도록 허용합니다. 자세한 내용은 [reference:admin/setup-wizard.pdf](#) 및 **Administration > Custom-channels**를 참조하십시오.
6. 필요한 하위 채널을 선택합니다(예: 필수 SUSE Manager 도구 및 업데이트 채널).
7. 옵션을 활성화하려면 **추가 시스템 유형** 확인란을 선택합니다.
8. **연결 방법**은 **기본값**으로 설정된 상태 그대로 두는 것이 좋습니다.
9. **범용 기본값** 설정은 확인란이 선택되지 않은 상태 그대로 두는 것이 좋습니다.
10. **활성화 키 생성**을 클릭하여 활성화 키를 생성합니다.
11. **구성 파일 배포** 확인란을 선택하여 이 키에 대한 구성 관리를 활성화하고, **활성화 키 업데이트**를 클릭하여 이 변경

사항을 저장합니다.



- 구성 파일 배포** 확인란은 활성화 키를 생성한 후에야 표시됩니다. 구성 관리를 활성화해야 하는 경우 되돌아가 확인란을 선택하십시오.

2.3.1. 여러 활성화 키 결합

기존 클라이언트에서 부트스트랩 스크립트를 실행할 때 활성화 키를 결합할 수 있습니다. 키를 결합하면 시스템에서 설치된 것에 대해 더 많은 제어권을 확보할 수 있고 규모가 크거나 복잡한 환경에서 키가 중복되는 일이 줄어듭니다.



- 활성화 키 결합은 기존 클라이언트에서만 작동합니다. Salt 클라이언트는 결합된 활성화 키를 지원하지 않습니다. Salt 클라이언트에서 결합된 키를 사용하는 경우 첫 번째 키만 사용됩니다.

명령 프롬프트나 단일 자동 설치 프로파일에서 여러 개의 활성화 키를 지정할 수 있습니다.

Uyuni 서버의 명령 프롬프트에서 **rhnreg_ks** 명령을 사용하고, 키 이름을 쉼표로 구분하십시오. Kickstart 프로파일에서 여러 개의 키를 지정하려면 **시스템** > **자동 설치**로 이동하여 사용하려는 프로파일을 편집하십시오.

일부 값들 사이의 충돌로 인해 클라이언트 등록이 실패할 수 있으므로 활성화 키를 결합할 때 주의하십시오. 시작하기 전에 다음 값에 서로 충돌하는 정보가 있는지 확인하십시오.

- 소프트웨어 패키지
- 소프트웨어 하위 채널
- 구성 채널

충돌이 감지되면 다음과 같이 처리됩니다.

- 기본 소프트웨어 채널의 충돌: 등록에 실패합니다.
- 시스템 유형의 충돌: 등록에 실패합니다.
- 구성 활성화** 플래그의 충돌: 구성 관리가 활성화됩니다.
- 한 개의 키가 시스템 전용인 경우: 등록에 실패합니다.

2.3.2. 활성화 키

클라이언트를 다시 등록하고 모든 Uyuni 설정을 다시 확보하기 위한 목적으로 재활성화 키를 한 번만 사용할 수 있습니다. 재활성화 키는 클라이언트 전용이며 시스템 ID, 이력, 그룹 및 채널을 포함합니다.

재활성화 키를 생성하려면 **시스템**으로 이동한 후 클라이언트를 클릭하여 재활성화 키를 생성하고, **상세 정보** > **재활성** 탭으로 이동하십시오. **새 키 생성**을 클릭하여 재활성화 키를 생성합니다. 나중에 사용할 수 있도록 키의 상세 정보를 기록해 둡니다. 특정 시스템 ID와 연결되지 않은 일반적인 활성화 키와 달리 여기에서 생성한 키는 **시스템** > **활성화 키** 페이지에 표시되지 않습니다.

Salt 클라이언트의 경우 재활성화 키를 생성한 후 이 키를 **management_key** (`/etc/salt/minion.d/susemanager.conf`)에서 입자로 사용할 수 있습니다. 예:

```
grains:
```

```
susemanager:
```

```
management_key: "re-1-daf44db90c0853edbb5db03f2b37986e"
```

재활성화 키를 적용하려면 **salt-minion** 프로세스를 다시 시작하십시오.

재활성화 키를 부트스트랩 스크립트와 함께 사용할 수 있습니다. 부트스트랩 스크립트에 대한 자세한 내용은 **Client-configuration > Registration-bootstrap**을 참조하십시오.

기존 클라이언트의 경우 재활성화 키를 생성한 후 이 키를 **rhnreg_ks** 명령줄 유ти리티와 함께 사용할 수 있습니다. 이 명령은 클라이언트를 다시 등록하고 Uyuni 설정을 복원합니다. 기존 클라이언트에서는 재활성화 키를 활성화 키와 결합하여 단일 시스템 프로파일에 대해 여러 키의 설정을 집계할 수 있습니다. 예:

```
rhnreg_ks --server=<server-url>/XMLRPC \
--activationkey=<reactivation-key>,<activationkey> \
--force
```



클라이언트를 기존 Uyuni 프로파일과 함께 자동 설치하는 경우 프로파일은 재활성화 키를 사용해 시스템을 다시 등록하고 설정을 복원합니다. 프로파일 기반 자동 설치가 진행 중일 때는 이 키를 재생성, 삭제 또는 사용하지 마십시오. 그러면 자동 설치에 실패하게 됩니다.

2.3.3. 활성화 키 모범 사례

기본 상위 채널

SUSE Manager 기본값 상위 채널을 사용하지 마십시오. 이 설정은 Uyuni가 설치된 운영 체제와 가장 부합하는 상위 채널을 선택하도록 강제합니다. 이로 인해 때로 예기치 않은 동작이 발생할 수 있습니다. 대신에 각 배포 및 아키텍처별로 활성화 키를 생성하는 것이 좋습니다.

활성화 키를 이용한 부트스트래핑

부트스트랩 스크립트를 사용 중인 경우 각 스크립트에 대해 활성화 키를 생성하는 것을 고려하십시오. 이렇게 하면 채널 할당, 패키지 설치, 시스템 그룹 구성원, 구성 채널 할당을 조정하는 데 도움이 됩니다. 또한 등록 후 시스템에 대한 수동 개입이 덜 필요합니다.

대역폭 요구사항

활성화 키를 사용하면 등록 시점에 소프트웨어가 자동으로 다운로드될 수 있는데, 대역폭이 제한된 환경에서는 바람직하지 않을 수 있습니다.

이러한 옵션은 대역폭 사용량을 생성합니다.

- SUSE 제품 풀 채널을 할당하면 해당 제품 설명자 패키지가 자동으로 설치됩니다.
- 섹션의 모든 패키지가 **Packages** 설치됩니다.
- **구성** 섹션의 모든 Salt 상태가 그 내용에 따라 다운로드를 트리거할 수 있습니다.

키 레이블 명명

활성화 키에 대해 읽을 수 있는 이름을 입력하지 않으면 시스템이 수 스트링을 자동으로 생성하므로 키를 관리하기 어려울 수 있습니다.

키를 추적하는 데 도움이 되는 활성화 키 명명 스키마를 고려하십시오. 조직의 인프라와 관련이 있는 이름을 생성하면 더 복잡한 작업을 더 쉽게 수행할 수 있습니다.

키 레이블을 생성할 때 다음과 같은 팁을 참고하십시오.

- OS 명명(필수): 키는 항상 자신이 설정을 제공하는 대상인 OS를 참조해야 합니다.
- 아키텍처 명명(권장): 귀사가 단 하나의 아키텍처(예: x86_64)만 운영하고 있지 않다면 레이블에 아키텍처 유형을 제공하는 것이 좋습니다.
- 서버 유형 명명: 이 서버를 어떤 목적으로 사용 중입니까?
- 위치 명명: 서버가 어디에 있습니까? 서버룸, 건물 또는 부서 내?
- 날짜 명명: 유지보수 기간, 분기 등
- 사용자 정의 명명: 어떤 명명 스키마가 조직의 필요에 적합합니까?

활성화 키 레이블 이름의 예:

```
sles15-sp4-web_server-room_129-x86_64
```

```
sles15-sp4-test_packages-blg_502-room_21-ppc64le
```

포함된 채널

활성화 키를 생성할 때는 어떤 소프트웨어 채널이 연결되어 있는지도 기억해 두어야 합니다. 키에는 할당된 특정 기본 채널이 있어야 합니다. 기본값으로 설정된 기본 채널은 사용하지 않는 것이 좋습니다. 자세한 내용은 **Client-configuration > Registration-overview**에서 설치하려는 클라이언트 운영 체제를 참조하십시오.

2.4. GPG 키

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.

대부분의 경우 GPG 설정을 조정하지 않아도 클라이언트에 소프트웨어를 설치할 수 있습니다.

RPM 패키지에는 직접 서명할 수 있지만, Debian 기반 시스템은 메타데이터에만 서명하고 체크섬 체인을 사용하여 패키지를 보호합니다. 대부분의 RPM 기반 시스템은 서명된 패키지뿐만 아니라 서명된 메타데이터를 사용합니다.

2.4.1. 클라이언트의 GPG 키 신뢰

운영 체제는 자체 GPG 키를 직접 신뢰하거나 최소 시스템 이상과 함께 설치된 상태로 제공됩니다. 그러나 다른 GPG 키로 서명된 타사 패키지는 수동으로 처리해야 합니다. 클라이언트는 신뢰할 수 있는 GPG 키가 없어도 성공적으로 부트스트래핑할 수 있습니다. 하지만 키를 신뢰할 수 있을 때까지는 새 클라이언트 도구 패키지를 설치하거나 업데이트할 수 없습니다.

Salt clients now use GPG key information entered for a software channel to manage the trusted keys. When a software channel with GPG key information is assigned to a client, the key is trusted as soon as the channel is refreshed or the first package gets installed from this channel.

The GPG key URL parameter in the software channel page can contain multiple key URLs separated by "whitespace". In case it is a file URL, the GPG key file must be deployed on the client before the software channel is used.

The GPG keys for the Client Tools Channels of Red Hat based clients are deployed on the client into `/etc/pki/rpm-gpg/` and can be referenced with file URLs.

Only in case a software channel is assigned to the client they will be imported and trusted by the system.



Debian 기반 시스템은 메타데이터에만 서명하므로 단일 채널에 대해 추가 키를 지정할 필요가 없습니다. **Administration > Repo-metadata**에서 "자체 GPG 키 사용"의 설명과 같이 사용자가 자체 GPG 키를 구성하여 메타데이터에 서명하는 경우 해당 키의 배포 및 신뢰가 자동으로 실행됩니다.

2.4.1.1. 사용자 정의 GPG 키

사용자는 클라이언트에 배포할 자체 GPG 키를 정의할 수 있습니다.

일부 열 데이터를 제공하고 Salt 파일 시스템에 GPG 키 파일을 제공하면 클라이언트에 자동으로 배포됩니다.

이러한 키는 RPM 기반 운영 체제의 `/etc/pki/rpm-gpg/` 및 Debian 시스템의 `/usr/share/keyrings/`에 배포됩니다.

키를 배포할 클라이언트에 대한 열 키 `custom_gpgkeys`를 정의하고 키 파일의 이름을 나열합니다.

```
cat /srv/pillar/mypillar.sls
custom_gpgkeys:
  - my_first_gpg.key
  - my_second_gpgkey.gpg
```

추가적으로 Salt 파일 시스템에서 `gpg` 디렉토리를 생성한 후 해당 디렉토리에 `custom_gpgkeys` 열 데이터에 지정된 이름으로 GPG 키 파일을 저장합니다.

```
ls -la /srv/salt/gpg/
/srv/salt/gpg/my_first_gpg.key
/srv/salt/gpg/my_second_gpgkey.gpg
```

이제 키는 `/etc/pki/rpm-gpg/my_first_gpg.key` 및 `/etc/pki/rpm-gpg/my_second_gpgkey.gpg`의 클라이언트에 배포됩니다.

마지막 단계로 소프트웨어 채널의 GPG 키 URL 필드에 URL을 추가합니다. **소프트웨어 > 관리 > 채널**로 이동하고 수정할 채널을 선택합니다. **GPG 키 URL**에 `file:///etc/pki/rpm-gpg/my_first_gpg.key` 값을 추가합니다.

2.4.1.2. 부트스트랩 스크립트의 GPG 키

절차: 부트스트랩 스크립트를 사용하여 클라이언트의 GPG 키 신뢰

- Uyuni 서버의 명령 프롬프트에서 `/srv/www/htdocs/pub/` 디렉토리의 내용을 확인합니다. 이 디렉토리에는 사용

가능한 모든 공용 키가 포함되어 있습니다. 등록하려는 클라이언트에 할당된 채널에 적용되는 키를 적어 두십시오.

- 관련 부트스트랩 스크립트를 열어 **ORG_GPG_KEY=** 파라미터를 찾고 필요한 키를 추가합니다. 예:

```
uyuni-gpg-pubkey-0d20833e.key
```

이전에 저장한 키는 삭제하지 않아도 됩니다.



- 클라이언트의 보안을 위해 GPG 키의 신뢰성이 중요합니다. 어떤 키가 필요하고 신뢰할 수 있는지 결정하는 것은 관리자가 수행해야 하는 작업입니다. GPG 키를 신뢰할 수 없으면 클라이언트에 소프트웨어 채널을 할당할 수 없습니다.

Chapter 3. 클라이언트 관리 방법

There are a number of ways that the Uyuni Server can communicate with clients. Which one you use depends on the type of client, and your network architecture:

Salt

is the default choice and recommended unless there are specific needs. For more information, see [contact-methods-salt.pdf](#).

Salt SSH

is useful only if network restrictions make it impossible for clients to establish contact to the server. This contact method has serious limitations. For more information, see [contact-methods-saltssh.pdf](#).

Traditional

is available for backwards compatibility only. This contact method has serious limitations. It does not scale as well as Salt.



Newer operating systems are not supported and will not be added in the future. The traditional contact method is deprecated and will be removed in the next version. Use it only when a needed feature is still not covered by Salt. For feature comparison, see [Client-configuration > Supported-features](#).

For more information, see [Client-configuration > Contact-methods-traditional](#).

3.1. Salt 클라이언트를 위한 연결 방법

The Salt contact method is the default choice and recommended unless there are specific needs. For more information about Salt in general, see [Specialized-guides > Salt](#).

The Salt Contact Method is the best scaling method. All new Uyuni features are supported and it has the widest variety of supported operating systems. All new operating systems are always supported with this contact method.

Software updates are generally pushed from the server to the client. Connections are initiated from the client. This means you must open ports on the server, not on clients. The Salt clients are also known as Salt minions. Uyuni Server installs a daemon on every client.

If you need to use Salt clients in a disconnected setup you can configure Push via Salt SSH as a contact method. With this contact method, clients can be located in a firewall-protected zone called a DMZ. For more information about Push via Salt SSH, see [Client-configuration > Contact-methods-saltssh](#).

3.1.1. 온보딩 정보

Salt에는 minion용 키를 저장하기 위한 자체 데이터베이스가 있습니다. 이것은 Uyuni 데이터베이스와 동기화된 상태로 유지되어야 합니다. Salt에서 키가 수락되면 Uyuni에서 온보딩 프로세스가 바로 시작됩니다. 온보딩 프로세스는 **minion_id** 및 **machine-id**를 검색하여 Uyuni 데이터베이스에서 기존 시스템을 찾습니다. 검색 결과가 없으면 새 시스템이 생성됩니다. **minion_id** 또는 **machine-id**가 포함된 항목이 있는 경우, 새 시스템과 일치하도록 해당 시스템이

マイグ레이션됩니다. 두 항목 모두에 대한 일치 항목이 있고 동일한 시스템이 아닌 경우, 오류와 함께 온보딩이 중단됩니다. 이 경우 관리자가 시스템을 1개 이상 제거하여 충돌을 해결해야 합니다.

3.1.2. Salt SSH를 통한 푸시

Push via Salt SSH is used in environments where Salt clients cannot reach the Uyuni Server directly. In this environment, clients are located in a firewall-protected zone called a DMZ. No system within the DMZ is authorized to open a connection to the internal network where the Uyuni Server is located.

Push via Salt SSH is also to use if installing a daemon agent on clients is not possible.

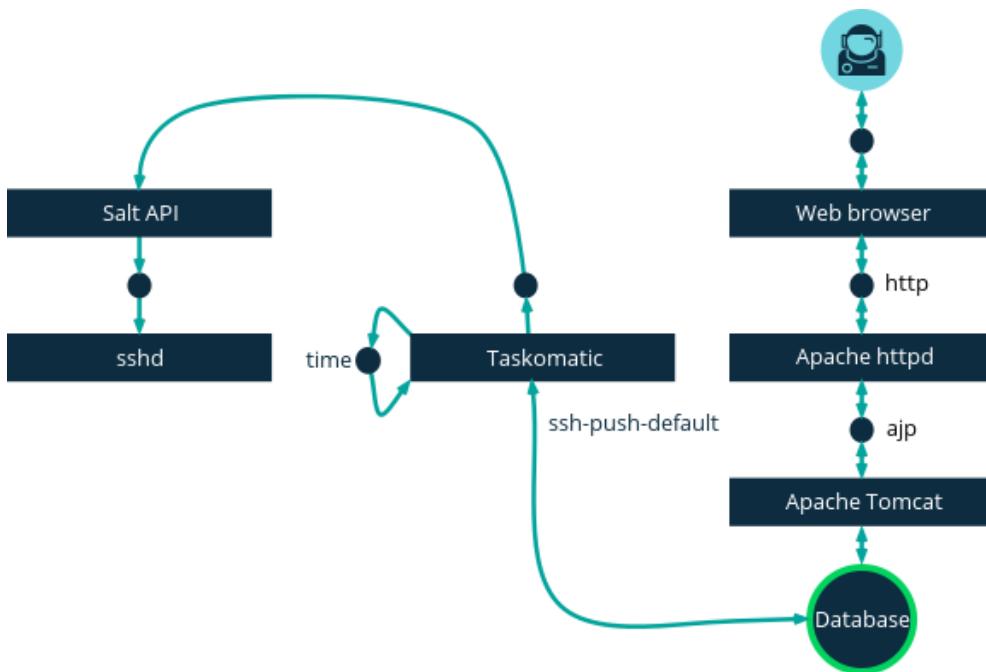


The Push via Salt SSH method has serious limitations. It does not scale well, and consumes more Server resources and network bandwidth than the plain Salt method. The Push via Salt SSH method is not at all supported with large setups (1000 clients and more).

The Push via Salt SSH method creates an encrypted tunnel from the Uyuni Server on the internal network to the clients located in the DMZ. After all actions such as updates and events are pushed and executed, the tunnel is closed.

The server uses the Salt SSH to contact the clients at regular intervals, checking in and performing scheduled actions and events.

아래 이미지는 Salt SSH를 통한 푸시 프로세스 경로를 나타낸 것입니다. Taskomatic 블록 왼쪽에 있는 모든 항목은 Uyuni 클라이언트에서 실행되는 프로세스를 나타냅니다.



To use Push via Salt SSH, you must have the SSH daemon running on the client, and reachable by the `salt-api` daemon running on the Uyuni Server. Additionally, the required Python version will be installed with the salt-bundle on the remote system.



Red Hat Enterprise Linux 5, CentOS 5 및 이전 버전은 지원되지 않는 Python 버전을 사용하므로 지원되지 않습니다.

절차: Salt SSH를 통한 푸시로 클라이언트 등록

1. Uyuni Web UI에서 **시스템**, **부트스트랩**으로 이동하여 해당 필드를 완성합니다.
2. Select an activation key with the Push via SSH contact method configured. For more information about activation keys, see **Client-configuration > Activation-keys**.
3. **SSH를 통해 시스템을 완전히 관리** 확인란을 선택합니다.
4. **부트스트랩**을 클릭하여 등록을 시작합니다.
5. **시스템**, **개요**로 이동하여 시스템이 올바르게 등록되었는지 확인합니다.

3.1.2.1. 사용 가능한 파라미터

Salt SSH를 통한 푸시를 구성하는 경우 호스트, 활성화 키, 비밀번호 등 시스템 등록 시 사용되는 파라미터를 수정할 수 있습니다. 비밀번호는 부트스트래핑에만 사용되며 어느 곳에도 저장되지 않습니다. 향후 모든 SSH 세션은 키/인증서 페어를 통해 인증됩니다. 이러한 파라미터는 **시스템** > **부트스트랩**에서 구성합니다.

You can also configure persistent parameters that are used system-wide, including the sudo user. For more information on configuring the sudo user, see **Client-configuration > Contact-methods-pushssh**.

3.1.2.2. 작업 실행

Salt SSH를 통한 푸시 기능은 Taskomatic을 사용해 **salt-ssh**로 예약된 작업을 실행합니다. Taskomatic 작업은 예약된 작업을 주기적으로 확인하여 실행합니다. 기존 클라이언트에서 SSH를 통해 푸시하는 것과 달리 Salt SSH를 통한 푸시 기능은 예약된 작업에 근거하여 완전한 **salt-ssh** 호출을 실행합니다.

기본적으로 한 번에 20건의 Salt SSH 작업을 실행할 수 있습니다. 구성 파일에 다음과 같은 줄을 추가하고 **parallel_threads**의 값을 상향 조정하여 병렬로 실행할 수 있는 작업의 수를 늘릴 수 있습니다. 문제를 방지하기 위해 병렬 작업의 수를 다음과 같이 낮게 유지하는 것이 좋습니다.

```
taskomatic.com.redhat.rhn.taskomatic.task.SSHMinionActionExecutor.parallel_threads = <number>
org.quartz.threadPool.threadCount = <value of parallel_threads + 20>
```

이렇게 하여 어느 한 곳의 클라이언트에서 병렬로 실행할 수 있는 작업의 수와 Taskomatic이 사용하는 작업자 스레드의 총 개수를 조정할 수 있습니다. 작업을 여러 클라이언트에서 실행해야 하는 경우 작업은 항상 각 클라이언트에서 순차적으로 실행됩니다.

클라이언트가 프록시를 통해 연결되어 있는 경우 프록시에서 **MaxSessions** 설정을 조정해야 합니다. 이 경우 병렬 연결의 수를 클라이언트 총 수의 세 배로 설정합니다.

3.1.2.3. 향후 출시될 기능

Salt SSH를 통한 푸시에서는 아직 지원하지 않는 몇 가지 기능이 있습니다. 다음 기능은 Salt SSH 클라이언트에서 작동하지 않습니다.

- OpenSCAP 감사
- Beacons, 이로 인해

- `zypper`를 사용해 시스템에 패키지를 설치해도 패키지 새로 고침이 호출되지 않습니다.
- 가상 호스트 기능(예: 게스트에 대한 호스트)은 가상 호스트 시스템이 Salt SSH 기반인 경우 작동하지 않습니다.

For more information about Salt SSH, see [Specialized-guides > Salt](#) and <https://docs.saltstack.com/en/latest/topics/ssh/>.

3.1.3. Salt Bundle

3.1.3.1. Salt Bundle이란 무엇입니까?

Salt Bundle은 Salt Minion, Python 3, 필수 Python 모듈 및 라이브러리가 포함된 단일 바이너리 패키지입니다.

Salt Bundle은 Python 3 및 Salt를 실행하기 위한 모든 요구 사항과 함께 제공됩니다. 따라서 Salt Bundle은 클라이언트에 설치된 Python 버전을 시스템 소프트웨어로 사용하지 않습니다. Salt Bundle은 해당 Salt 버전에 대한 요구 사항을 충족하지 않는 클라이언트에 설치할 수 있습니다.

Uyuni Salt Master가 아닌 Salt Master에 연결된 Salt Minion을 실행하는 시스템에서도 Salt Bundle을 사용할 수 있습니다.

3.1.3.2. Salt Bundle을 Minion으로 클라이언트 등록

Salt Bundle에 등록하는 방법이 권장되는 등록 방법입니다. 이 섹션에서는 현재 구현의 장점과 제한 사항을 설명합니다. Salt Bundle은 Salt, Python 3 및 Salt가 사용하는 Python 모듈로 구성된 `venv-salt-minion`으로 제공됩니다. Web UI를 사용한 부트스트랩은 Salt Bundle도 사용하므로 Web UI를 사용한 부트스트랩은 Python에 종속되지 않습니다. Salt Bundle을 사용하면 더 이상 클라이언트가 Python 해석기 또는 모듈을 제공할 필요가 없습니다.

새 클라이언트를 부트스트랩하는 경우 Salt Bundle에 등록하는 것이 기본 방법입니다. 기존 클라이언트를 Salt Bundle 방식으로 전환할 수 있습니다. 전환하면 `salt-minion` 패키지와 종속 항목이 설치된 상태로 유지됩니다.

3.1.3.2.1. Salt Minion과 함께 Salt Bundle 사용

Salt Bundle은 Uyuni Server가 아닌 Salt Master가 관리하는 Salt Minion과 동시에 사용할 수 있습니다. Salt Bundle이 클라이언트에 설치된 경우 Uyuni Server가 Salt Bundle의 구성 파일을 관리하며, 이 경우 `salt-minion`의 구성 파일은 관리되지 않습니다. 자세한 내용은 [Salt Bundle 구성](#)을 참조하십시오.



- Uyuni Server가 아닌 Salt Master가 관리하는 Salt Minion으로 클라이언트를 부트스트랩하려면 부트스트랩을 생성할 때 `mgr-bootstrap --force-bundle` 을 사용하는 것이 좋습니다. 스크립트를 사용하거나 부트스트랩 스크립트에서 `FORCE_VENV_SALT_MINION` 을 1로 설정하십시오.
- Web UI `mgr_force_venv_salt_minion` 이 `true` 로 설정된 부트스트래핑의 경우 열을 전역적으로 지정할 수 있습니다. 자세한 내용은 [Specialized-guides > Salt](#)에서 확인할 수 있습니다.

3.1.3.2.2. Salt Minion에서 Salt Bundle로 전환

Salt 상태 `util.mgr_switch_to_venv_minion` 을 사용하여 `salt-minion` 에서 `venv-salt-minion` 으로 전환할 수 있습니다. 프로세스 이동 문제를 방지하려면 두 단계에 걸쳐 `venv-salt-minion` 으로 전환하는 것이 좋습니다.

절차: `util.mgr_switch_to_venv_minion` 상태를 `venv-salt-minion` 으로 전환

1. 우선 열을 지정하지 않고 `util.mgr_switch_to_venv_minion` 을 적용하십시오. 그러면 구성 파일 등을 복사하는 `venv-salt-minion` 으로 전환됩니다. 원래 `salt-minion` 구성 및 해당 패키지는 정리되지 않습니다.

```
salt <minion_id> state.apply util.mgr_switch_to_venv_minion
```

`mgr_purge_non_venv_salt` 가 `True` 로 설정된 `util.mgr_switch_to_venv_minion` 을 적용하여 `salt-minion` 을 제거하고 `mgr_purge_non_venv_salt_files` 를 `True` 로 설정하여 `salt-minion` 과 관련된 모든 파일을 제거하십시오. 이 두 번째 단계는 첫 번째 단계가 처리되었는지 확인한 후 이전 구성 파일과 더 이상 사용되지 않는 `salt-minion` 패키지를 제거합니다.

```
salt <minion_id> state.apply util.mgr_switch_to_venv_minion
pillar='{"mgr_purge_non_venv_salt_files": True,
"mgr_purge_non_venv_salt": True}'
```



- 첫 번째 단계를 건너뛰고 두 번째 전환 단계를 실행하는 경우 클라이언트 측에서 명령을 실행하기 위해 사용되는 `salt-minion` 을 중지해야 하므로 상태 적용 프로세스가 실패할 수 있습니다.

반면에 Salt Bundle을 설치하지 않고 대신 `salt-minion` 을 계속 사용하는 것도 가능합니다. 이 경우 다음 옵션 중 하나를 지정:

- `--no-bundle` 옵션으로 `mgr-bootstrap` 을 실행하십시오.
- 생성된 부트스트랩 스크립트에서 `AVOID_VENV_SALT_MINION` 을 `1` 로 설정합니다.
- 부트스트랩 상태에 대하여 `mgr_avoid_venv_salt_minion` 열을 `True` 로 설정합니다.

3.1.3.3. Salt Bundle을 사용한 Salt SSH

Salt Bundle은 클라이언트로 Salt SSH 작업을 수행할 때도 사용됩니다.

셀 스크립트는 Salt 명령이 실행되기 전 `venv-salt-minion` 을 설치하지 않고 대상 시스템에 Salt Bundle을 배포합니다. Salt Bundle에는 전체 Salt 코드 기반이 포함되어 있으므로 `salt-thin` 이 배포되지 않습니다. Salt SSH(Web UI를 사용한 부트스트랩 포함)는 번들 내의 Python 3 해석기를 사용합니다. 대상 시스템에는 다른 Python 해석기를 설치할 필요가 없습니다.

Bundle과 함께 배포된 Python 3는 클라이언트에서 Salt SSH 세션을 처리하기 위해 사용되므로 Salt SSH(Web UI를 사용한 부트스트래핑 포함)는 시스템에 설치된 Python에 종속되지 않습니다.

`salt-thin` 을 사용하는 것은 대체 방법으로 활성화할 수 있지만 클라이언트에 Python 3을 설치해야 합니다. 이 방법은 권장되거나 지원되지 않으며 개발 목적으로만 존재합니다. `/etc/rhn/rhn.conf` 구성 파일에서 `web.ssh_use_salt_thin` 을 `true` 로 설정합니다.



- The bootstrap repository must be created before bootstrapping the client with Web UI. Salt SSH is using the Salt Bundle taken from the bootstrap repository based on the detected target operating system. For more information, see [client-configuration:bootstrap-repository.pdf](#).

- Salt SSH는 `/var/tmp` 를 사용하여 Salt Bundle을 배포하고 번들된 Python으로 클라이언트에서 Salt 명령을 실행합니다. 그러므로 `noexec` 옵션으로 `/var/tmp` 를 마운트하지 않아야 합니다. 부트스트랩 프로세스는 Salt SSH를 사용하여 클라이언트에 도달하므로 `/var/tmp` 가 `noexec` 옵션으로 마운트된 클라이언트를 Web UI로 부트스트랩할 수 없습니다.

3.1.3.4. pip를 사용하여 Python 패키지와 함께 Salt Bundle 확장

Salt Bundle에는 번들로 제공되는 Salt Minion의 기능을 추가 Python 패키지로 확장할 수 있는 `pip` 가 포함되어 있습니다.

기본적으로 `salt <minion_id> pip.install <package-name>` 은 `<package_name>` 에 의해 지정된 Python 패키지를 `/var/lib/venv-salt-minion/local` 에 설치합니다.

 필요한 경우 `/var/lib/venv-salt-minion/local` 경로는 `venv-salt-minion.service` 에 대한 `VENV_PIP_TARGET` 환경 변수를 설정하여 재정의할 수 있습니다. 서비스에 대한 systemd drop-in 구성 파일을 사용하는 것이 좋습니다. 이는 구성 파일 `/etc/systemd/system/venv-salt-minion.service.d/10-pip-destination.conf` 를 사용하여 수행할 수 있습니다.

```
[Service]
Environment=VENV_PIP_TARGET=/new/path/local/venv-salt-
minion/pip
```

 `pip` 를 통해 설치된 Python 패키지는 Salt Bundle을 업데이트할 때 변경되지 않습니다. 업데이트 이후에 해당 패키지를 사용 가능하고 작동하는 상태로 만들려면 Salt Bundle 업데이트 후에 적용되는 Salt 상태로 패키지를 설치하는 것이 좋습니다.

3.2. 기존 클라이언트를 위한 연결 방법

기존 클라이언트는 다양한 방법을 사용해 Uyuni 서버와 통신할 수 있습니다.

The lightweight Uyuni daemon (`rhnscd`) runs on traditional client systems. The daemon periodically connects with Uyuni to check for new updates and notifications.



The `rhnscd` method has serious limitations. It does not scale as well as Salt. Newer operating systems are not supported and will not be added in the future.

For more information, see [Client-configuration > Contact-methods-rhnscd](#).

Traditional with OSAD

is the same as traditional but allows the server to push updates to clients. OSAD is an enhancement to `rhnscd`. OSAD allows traditional clients to execute scheduled actions immediately. It does not apply to Salt clients.

Traditional SSH Push

is same as traditional but allows the server to push updates to clients, using the SSH protocol as a transport layer.

SSH를 통한 푸시는 클라이언트가 Uyuni 서버에 직접 도달할 수 없는 환경에서 사용됩니다. 이 환경에서 클라이언트는 방화벽으로 보호되는 영역인 DMZ라는 곳에 있습니다. DMZ 내의 어떤 시스템도 Uyuni 서버를 포함한 내부 네트워크에 연결할 권한이 없습니다.

Traditional SSH Push with Tunnel

The same as SSH Push, but tunnels HTTP/HTTPS traffic (for package download) via SSH.

SUSE Manager 4.3 릴리스에서는 기존 클라이언트를 더 이상 사용하지 않습니다. SUSE Manager 4.3 이후의 릴리스는 기존 클라이언트 및 기존 프록시를 지원하지 않으며, 이는 2023년으로 예정되어 있습니다. 모든 신규 배포에서 Salt 클라이언트와 Salt 프록시만 사용하고 기존 클라이언트와 프록시를 Salt로 마이그레이션하는 것이 좋습니다.



Be aware that when migrating from traditional clients to Salt minions you do not have to delete the registered clients before. You can just register them as Salt minions and Salt will do the necessary steps with the traditional client. If you already deleted the traditional client and the registration as Salt minion is not possible anymore, see **Administration > Troubleshooting**.

3.2.1. SUSE Manager 데몬(rhnsd)

Uyuni 데몬(**rhnsd**)은 기존 클라이언트 시스템에서 실행되며 Uyuni와 주기적으로 연결하여 새로운 업데이트 및 알림을 확인합니다. Salt 클라이언트에는 적용되지 않습니다.

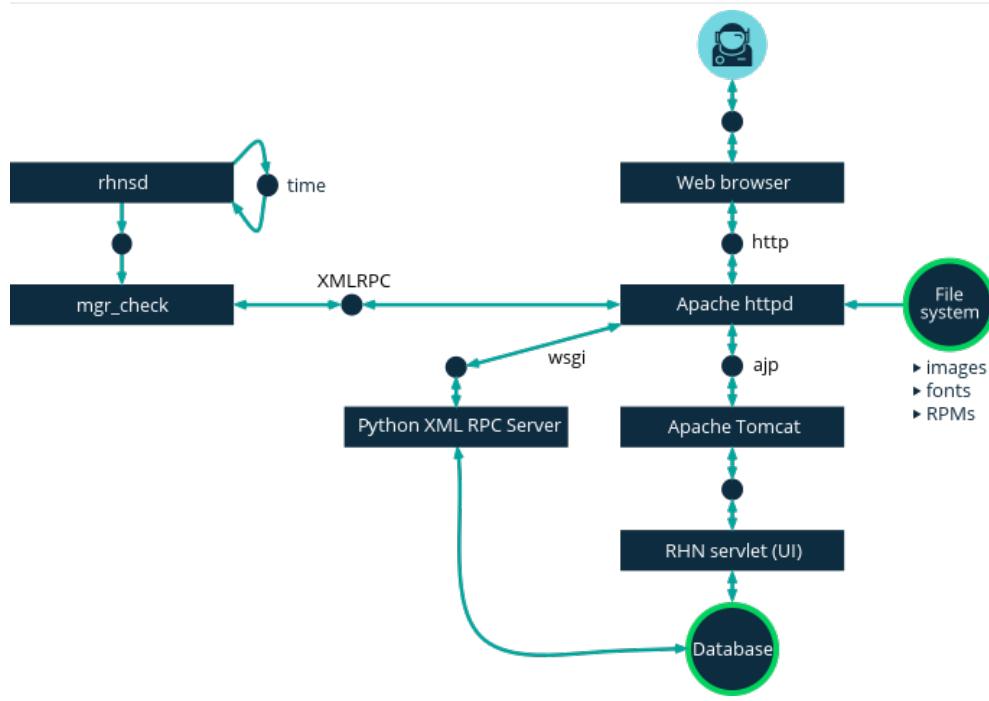
3.2.1.1. Start rhnsd

A systemd timer (**rhnsd.timer**) is used and controlled by **rhnsd.service**.

By default, **rhnsd** checks every four hours for new actions. This means it can take some time for clients to execute scheduled actions.

업데이트를 확인하기 위해 **rhnsd**는 `/usr/sbin/`에 있는 외부 **mgr_check** 프로그램을 실행합니다. Uyuni에 대한 네트워크 연결을 설정하는 소형 응용 프로그램입니다. Uyuni 데몬은 네트워크 포트를 수신 대기하거나 네트워크와 직접 대화하지 않습니다. 모든 네트워크 활동은 **mgr_check** 유ти리티가 수행합니다.

이 그림은 기본 **rhnsd** 프로세스 경로의 개요를 제공합니다. **Python XMLRPC 서버** 블록 왼쪽에 있는 모든 항목은 Uyuni 클라이언트에서 실행되는 프로세스를 나타냅니다.



3.2.1.2. rhnsd 구성

SUSE Linux Enterprise 12 이후에서는 기본 시간 간격이 다음 섹션과 같이 `/etc/systemd/system/timers.target.wants/rhnsd.timer`에 설정됩니다.

```
[Timer]
OnCalendar=00/4:00
RandomizedDelaySec=30min
```

다음과 같이 `systemctl`을 사용해 `rhnsd.timer`에 대해 무효화 드롭인 파일을 생성할 수 있습니다.

```
systemctl edit rhnsd.timer
```

예를 들어 시간 간격을 두 시간을 구성하고 싶은 경우 다음과 같이 합니다.

```
[Timer]
OnCalendar=00/2:00
```

파일은 `/etc/systemd/system/rhnsd.timer.d/override.conf`로 저장됩니다.

`systemd` 타이머에 대한 자세한 내용은 `systemd.timer` and `systemctl` 맨 페이지를 확인하십시오.

3.2.1.3. OSAD

OSAD는 Uyuni와 기존 클라이언트 간 연락 방법을 대체할 수 있는 수단입니다. 기본적으로 Uyuni는 네 시간마다 서버에 연결하여 예약된 작업을 실행하는 `rhnsd`를 사용합니다. OSAD를 통해 기존 클라이언트는 예약된 작업을 즉시 실행할 수

있습니다.



- rhnsm** 뿐 아니라 OSAD도 사용하십시오. **rhnsm** 를 비활성화하면 24시간 후에 클라이언트가 체크인하지 않은 것으로 표시됩니다.

OSAD에는 서로 뚜렷이 구분되는 구성요소가 몇 가지 있습니다.

- **osa-dispatcher** 서비스는 서버에서 실행되며 데이터베이스 검사를 사용해 클라이언트에 ping을 수행해야 하는지 또는 작업을 실행해야 하는지 여부를 판단합니다.
- **osad** 서비스는 클라이언트에서 실행됩니다. 이 서비스는 **osa-dispatcher**에서 보내는 ping에 반응하며 **mgr_check** 을 실행하여 지시에 따라 작업을 수행합니다.
- **jabberd** 서비스는 클라이언트와 서버 간 통신을 위해 XMPP 프로토콜을 사용하는 데몬입니다. 또한 **jabberd** 서비스는 인증을 처리합니다.
- **mgr_check** 도구는 작업을 수행할 클라이언트에서 실행됩니다. **osa-dispatcher** 서비스가 통신을 시도하면 트리거됩니다.

osa-dispatcher 는 주기적으로 쿼리를 실행하여 클라이언트가 마지막으로 네트워크 활동을 보인 시점을 확인합니다. 최근에 활동을 보이지 않은 클라이언트를 찾으면 Uyuni 서버에 등록된 모든 클라이언트에서 실행되는 모든 **osad** 인스턴스에 **jabberd** 를 사용해 ping을 실행합니다. **osad** 인스턴스는 서버의 배경에서 실행 중인 **jabberd** 를 사용해 ping에 응답합니다. **osa-dispatcher** 가 이 응답을 수신하면 클라이언트를 온라인으로 표시합니다. **osa-dispatcher** 가 특정 기간 내에 응답을 수신하는 데 실패하는 경우에는 클라이언트를 오프라인으로 표시합니다.

OSAD를 지원하는 시스템에서 작업을 예약하면 작업이 즉시 수행됩니다. **osa-dispatcher** 는 클라이언트에 실행해야 할 작업이 있는지 주기적으로 확인합니다. 미결 작업이 발견되면 **jabberd** 를 사용해 클라이언트에서 **mgr_check** 를 실행합니다. 그러면 클라이언트가 작업을 실행합니다.

OSAD 클라이언트는 서버의 전체 도메인 이름(FQDN)을 사용해 **osa-dispatcher** 서비스와 통신합니다.

SSL이 있어야 **osad** 통신이 가능합니다. SSL 인증서를 사용할 수 없는 경우 클라이언트 시스템의 데몬이 연결하지 못합니다. 방화벽 규칙이 필요한 포트를 허용하도록 설정되어 있는지 확인하십시오. 자세한 내용은 **Installation-and-upgrade > Ports**를 참조하십시오.

절차: OSAD 활성화

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **osa-dispatcher** 서비스를 시작합니다.

```
systemctl start osa-dispatcher
```

2. 각 클라이언트의 Tools 하위 채널에서 **mgr-osad** 패키지를 설치합니다. **mgr-osad** 패키지는 클라이언트에만 설치해야 합니다. Uyuni 서버에 **mgr-osad** 패키지를 설치하면 **osa-dispatcher** 패키지와 충돌합니다.

3. 각 클라이언트에서 루트 권한으로 다음과 같이 **osad** 서비스를 시작합니다.

```
systemctl start osad
```

osad 및 **osa-dispatcher** 가 서비스로 실행되므로 **stop**, **restart**, **status** 등 표준 명령을 사용해 관리할 수 있습니다.

각 OSAD 구성요소는 로컬 구성 파일을 사용해 구성합니다. 모든 OSAD 구성요소에 대해 기본 구성 파라미터를 유지하는 것이 좋습니다.

구성요소	위치	구성 파일 경로
osa-dispatcher	서버	/etc/rhn/rhn.conf 섹션: OSA configuration
osad	클라이언트	/etc/sysconfig/rhn/osad.conf
osad 로그 파일	클라이언트	/var/log/osad
jabberd 로그 파일	서버와 클라이언트 모두	/var/log/messages

OSAD 문제 해결

OSAD 클라이언트가 서버에 연결할 수 없거나 **jabberd** 서비스가 포트 5552에 응답하는 데 많은 시간이 걸리는 경우 이는 열린 파일 수를 초과했기 때문일 수 있습니다.

각 클라이언트에는 서버에 대한 한 건의 상시 개방 TCP 연결이 필요합니다. 이 연결에서는 하나의 파일 처리기를 사용합니다. 현재 개방된 파일 처리기의 수가 **jabberd** 사용이 허용되는 최대 파일 수를 초과하면 **jabberd** 가 요청을 대기열로 보내고 연결을 거부합니다.

이 문제를 해결하려면 **/etc/security/limits.conf** 구성 파일을 편집하고 다음 줄을 추가하여 **jabberd** 의 파일 한도를 높이면 됩니다.

```
jabber soft nofile 5100
jabber hard nofile 6000
```

소프트 한도의 경우 클라이언트 수에 100을 더하고, 하드 한도의 경우 현재 클라이언트 수에 1000을 더하여 환경에 필요한 한도를 계산합니다.

위 예에서는 현재 클라이언트를 500개로 가정하였으므로 소프트 한도는 5100이고 하드 한도는 6000입니다.

또한 선택한 하드 한도로 **/etc/jabberd/c2s.xml** 파일에서 다음과 같이 **max_fds** 파라미터를 업데이트해야 합니다.

```
<max_fds>6000</max_fds>
```

3.2.2. SSH를 통한 푸시

SSH를 통한 푸시는 기존 클라이언트가 Uyuni 서버에 직접 도달할 수 없는 환경에서 사용됩니다. 이 환경에서 클라이언트는 방화벽으로 보호되는 영역인 DMZ라는 곳에 있습니다. DMZ 내의 어떤 시스템도 Uyuni 서버를 포함한 내부 네트워크에 연결할 권한이 없습니다.

SSH를 통한 푸시 방법은 DMZ에 있는 클라이언트에 대한 내부 네트워크의 Uyuni 서버에서 암호화된 터널을 생성합니다. 모든 작업과 이벤트가 실행되고 나면 터널이 종료됩니다.

서버는 SSH를 사용해 정기적으로 클라이언트에 연결하여 체크인하고 예약된 작업과 이벤트를 수행합니다.

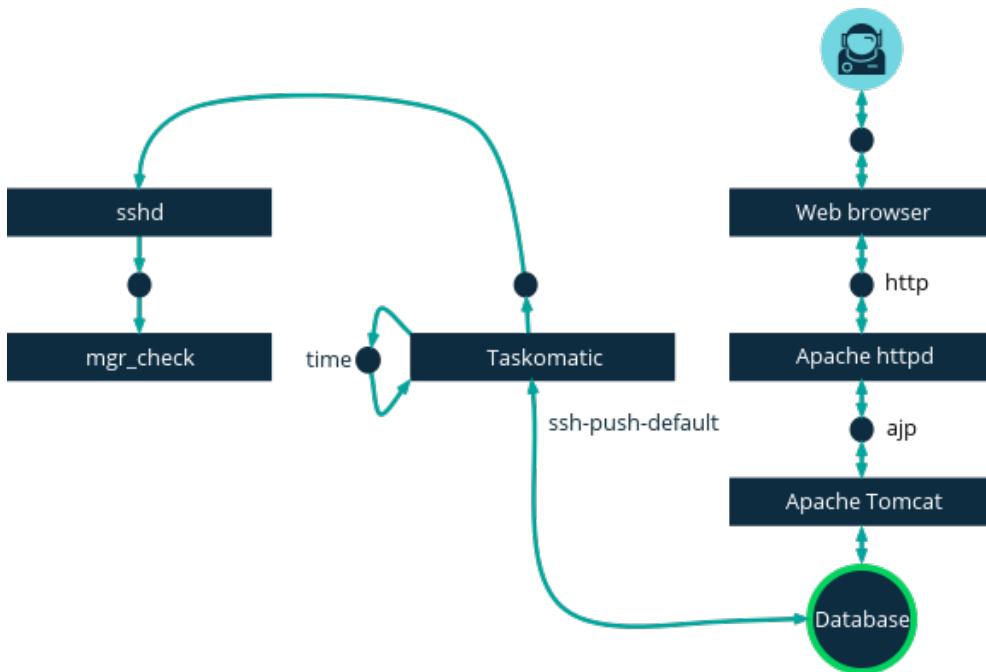
이 연결 방법은 기존 클라이언트에 대해서만 작동합니다. Salt 클라이언트에 대해서는 Salt SSH를 통한 푸시를

사용하십시오.



- 조달 모델을 사용한 시스템 재설치는 SSH를 통한 푸시로 관리되는 클라이언트에서는 현재 지원되지 않습니다.

이 이미지는 SSH를 통한 푸시 프로세스 경로를 나타낸 것입니다. Taskomatic 블록 왼쪽에 있는 모든 항목은 Uyuni 클라이언트에서 실행되는 프로세스를 나타냅니다.



SSH를 통한 터널링 연결의 경우 사용 가능한 포트 번호가 두 개 필요합니다. 하나는 터널링 HTTP를 위한 것이고 다른 하나는 HTTPS를 통한 터널링을 위한 것입니다(HTTP는 등록 프로세스 중에만 필요함). 기본적으로 사용되는 포트 번호는 1232 및 1233입니다. 이 번호를 덮어쓰려면 다음과 같이 1024보다 큰 사용자 정의 포트 번호 두 개를 /etc/rhn/rhn.conf에 추가하면 됩니다.

```
ssh_push_port_http = high_port_1
ssh_push_port_https = high_port_2
```

IP 주소 대신에 호스트 이름을 사용해 클라이언트에 연결하고 싶다면 다음 옵션을 설정하십시오.

```
ssh_push_use_hostname = true
```

클라이언트에 병렬로 연결하는 데 사용할 스레드의 수를 조정할 수도 있습니다. 기본적으로 두 개의 병렬 스레드를 사용합니다. /etc/rhn/rhn.conf에서 taskomatic.ssh_push_workers를 다음과 같이 설정합니다.

```
taskomatic.ssh_push_workers = number
```

보안상의 이유로 sudo를 SSH와 함께 사용해 루트 권한 대신에 권한이 없는 사용자로 시스템에 액세스하고 싶을 때가 있을 수 있습니다.

절차: 권한이 없는 SSH 액세스 구성

1. Uyuni 서버에 최신 **spacewalk-taskomatic** 및 **spacewalk-certs-tools** 패키지가 설치되어 있는지 확인합니다.
2. 각 클라이언트 시스템에서 적절한 권한 없는 사용자를 생성합니다.
3. 각 클라이언트 시스템에서 **/etc/sudoers** 파일을 열고 다음과 같은 줄을 코멘트 아웃합니다.

```
#Defaults targetpw    # ask for the password of the target user i.e.
root
#ALL      ALL=(ALL) ALL      # WARNING! Only use this together with
'Defaults  targetpw'!
```

4. 각 클라이언트 시스템의 **사용자 권한 사양** 섹션에서 다음과 같은 줄을 추가합니다.

```
<user> ALL=(ALL) NOPASSWD:/usr/sbin/mgr_check
<user> ALL=(ALL) NOPASSWD:/home/<user>/enable.sh
<user> ALL=(ALL) NOPASSWD:/home/<user>/bootstrap.sh
```

5. 각 클라이언트 시스템의 **/home/<user>/.bashrc** 파일에 다음과 같은 줄을 추가합니다.

```
PATH=$PATH:/usr/sbin
export PATH
```

6. Uyuni 서버의 **/etc/rhn/rhn.conf** 구성 파일에서 다음과 같은 줄을 추가하거나 수정하여 권한이 없는 사용자 이름을 포함합니다.

```
ssh_push_sudo_user = <user>
```

클라이언트가 DMZ에 있어 서버에 도달할 수 없으므로 **mgr-ssh-push-init** 도구를 사용해 Uyuni 서버에 클라이언트를 등록해야 합니다.

도구를 사용하려면 클라이언트 호스트 이름 또는 IP 주소와 Uyuni 서버의 유효한 부트스트랩 스크립트에 대한 경로가 필요합니다. 부트스트래핑에 대한 자세한 내용은 **Client-configuration** > **Registration-bootstrap**을 참조하십시오.

부트스트랩 스크립트에는 SSH를 통한 푸시에 대해 구성된 활성화 키가 연결돼 있어야 합니다. 활성화 키에 대한 자세한 내용은 **Client-configuration** > **Activation-keys**를 참조하십시오.

시작하기 전에 SSH 터널링에 사용할 포트를 지정했는지 확인해야 합니다. 포트 번호를 변경하기 전에 클라이언트를 등록한 경우 다시 등록해야 합니다.



- SSH를 통한 푸시로 관리되는 클라이언트는 서버에 직접 도달할 수 없습니다. **mgr-ssh-push-init** 도구를 사용하는 경우 **rhnsd** 데몬이 비활성화됩니다.

절차: SSH를 통한 푸시로 클라이언트 등록

- Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음 명령을 실행합니다.

```
# mgr-ssh-push-init --client <client> --register \
/srv/www/htdocs/pub/bootstrap/bootstrap_script --tunnel
```

옵션: 터널링을 사용하고 싶지 않다면 **--tunnel** 옵션을 제거하면 됩니다.

- 옵션: **ssh_push_sudo_user**를 정의했다면 **--notty** 옵션을 추가하여 루트 비밀번호 사용을 허용할 수 있습니다.
- 다음과 같이 SSH 연결이 활성화되어 있는지 확인합니다.

```
# ssh -i /root/.ssh/id_susemanager -R <high_port>:<susemanager>:443 \
<client> zypper ref
```

예: SSH를 통한 푸시에 대한 API 액세스

API를 사용해 어떤 연결 방법을 사용할지 관리할 수 있습니다. 다음 Python 코드 예제에서는 연결 방법을 **ssh-push**로 설정합니다.

유효한 값은 다음과 같습니다.

- default** (풀)
- ssh-push**
- ssh-push-tunnel**

```
client = xmlrpclib.Server(SUMA_HOST + '/rpc/api', verbose=0)
key = client.auth.login(SUMA_LOGIN, SUMA_PASSWORD)
client.system.setDetails(key, 1000012345, {'contact_method' : 'ssh-
push'})
```

이미 등록한 클라이언트가 있는데 이 클라이언트를 마이그레이션하여 SSH를 통한 푸시를 사용하고 싶다면 몇 가지 단계가 추가로 필요합니다. **mgr-ssh-push-init** 도구를 사용해 클라이언트를 설정할 수 있습니다.

절차: 등록한 시스템을 SSH를 통한 푸시로 마이그레이션

- Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 클라이언트를 설정합니다.

```
# mgr-ssh-push-init --client <client> \
/srv/www/htdocs/pub/bootstrap/bootstrap_script --tunnel
```

- Uyuni Web UI를 사용해 클라이언트의 연결 방법을 **ssh-push** 또는 **ssh-push-tunnel**로 변경합니다.
- 옵션: 기존 활성화 키를 편집해야 하는 경우 다음 명령을 사용할 수 있습니다.

```
client.activationkey.setDetails(key, '1-mykey', {'contact_method' :  
    'ssh-push'})
```

또한 프록시를 사용하여 연결하는 클라이언트에 대해 SSH를 통한 푸시를 사용할 수 있습니다. 시작하기 전에 프록시가 업데이트되었는지 확인하십시오.

절차: SSH를 통한 푸시로 클라이언트를 프록시에 등록

1. Uyuni 프록시의 명령 프롬프트에서 루트 권한으로 다음과 같이 클라이언트를 설정합니다.

```
# mgr-ssh-push-init --client <client> \  
/srv/www/htdocs/pub/bootstrap/bootstrap_script --tunnel
```

2. Uyuni 서버의 명령 프롬프트에서 다음과 같이 SSH 키를 프록시에 복사합니다.

```
mgr-ssh-push-init --client <proxy>
```

Chapter 4. 클라이언트 등록

클라이언트를 Uyuni 서버에 등록하는 방법이 몇 가지 있습니다. 이 섹션에서는 사용할 수 있는 다양한 방법에 대해 설명합니다. 또한 클라이언트에서 실행하려는 운영 체제에 대한 정보도 포함되어 있습니다.

시작하기 전에 다음 사항을 확인하십시오.

- 클라이언트는 등록 전에 날짜 및 시간을 Uyuni 서버와 정확하게 동기화한 상태이어야 합니다.
- 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.



Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

4.1. 클라이언트 등록 방법

클라이언트를 Uyuni 서버에 등록하는 방법이 몇 가지 있습니다.

- Salt 클라이언트의 경우 Uyuni Web UI를 사용해 클라이언트를 등록하는 것이 좋습니다. 자세한 내용은 [Client-configuration > Registration-webui](#)를 참조하십시오.
- 프로세스에 대한 제어 권한을 더 많이 갖고 싶거나 여러 클라이언트를 등록해야 하거나 기존 클라이언트를 등록하려면 부트스트랩 스크립트를 생성하는 것이 좋습니다. 자세한 내용은 [Client-configuration > Registration-bootstrap](#)에서 참조하십시오.
- Salt 클라이언트와 프로세스에 대한 더 많은 제어 권한을 위해서는 명령줄에서 단일 명령을 실행하면 도움이 될 수 있습니다. 자세한 내용은 [Client-configuration > Registration-cli](#)를 참조하십시오.

클라이언트는 등록 전에 날짜 및 시간을 Uyuni 서버와 정확하게 동기화한 상태이어야 합니다.

부트스트랩 스크립트 또는 명령줄 방법을 사용하려면 먼저 활성화 키를 생성해야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.



Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

4.1.1. Web UI로 클라이언트 등록

Uyuni Web UI로 클라이언트를 등록하는 방법은 Salt 클라이언트에만 사용할 수 있습니다.

Web UI를 사용하여 Salt 클라이언트를 부트스트랩하는 경우 클라이언트에서 부트스트랩 프로세스를 실행하기 위해 [Specialized-guides > Salt](#)를 사용하는 것입니다. Salt SSH는 Salt Bundle 및 포함된 Python 해석기를 사용합니다. 그러므로 클라이언트에 다른 Python 해석기를 설치할 필요가 없습니다.



Salt Bundle은 부트스트랩 리포지토리와 함께 제공되므로 리포지토리를 생성한 후 클라이언트의 부트스트랩 프로세스를 시작해야 합니다. 셸 스크립트가 클라이언트에서 운영 체제를 감지하고 부트스트랩 스크립트와 동일한 논리를 사용하여 적절한 부트스트랩 리포지토리로부터 Salt Bundle을 배포합니다. 자세한 내용은 [부트스트랩 리포지토리 생성](#)

준비를 참조하십시오.



- Do not register the Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

절차: Web UI로 클라이언트 등록

1. Uyuni Web UI에서 **시스템**, **부트스트랩**으로 이동합니다.
2. **호스트** 필드에 부트스트랩할 클라이언트의 정규화된 도메인 이름(FQDN)을 입력합니다.
3. **SSH 포트** 필드에서 클라이언트를 연결하고 부트스트랩하는 데 사용할 SSH 포트 번호를 입력합니다. 기본적으로 SSH 포트는 **22**입니다.
4. **사용자** 필드에서 클라이언트에 로그인할 사용자 이름을 입력합니다. 기본적으로 사용자 이름은 **root**입니다.
5. SSH로 클라이언트를 부트스트래핑하려면 **인증** 필드에서 **SSH 개인 키**를 확인하고 클라이언트에 로그인할 때 사용할 SSH 개인 키를 업로드하십시오. SSH 개인 키에 암호문이 필요한 경우 **SSH 개인 키 암호문** 필드에 입력하십시오. 또는 어떤 암호문에 대해서도 이 필드를 공백 상태로 두지 마십시오.
6. 비밀번호로 클라이언트를 부트스트래핑하려면 **인증** 필드에서 **비밀번호**를 확인하고 비밀번호를 입력하여 클라이언트에 로그인하십시오.
7. **활성화 키** 필드에서 클라이언트를 부트스트랩하는 데 사용할 소프트웨어 채널과 연결된 활성화 키를 선택합니다. 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.
8. 옵션: **프록시** 필드에서 클라이언트를 등록할 프록시를 선택합니다.
9. 기본적으로 **SSH Strict 키 호스트 점검 비활성화** 확인란이 선택되어 있습니다. 따라서 사용자가 수동으로 인증할 필요 없이 부트스트랩 프로세스가 자동으로 SSH 호스트 키를 수락할 수 있습니다.
10. 옵션: **SSH를 통해 시스템을 완전히 관리** 확인란을 선택합니다. 이 옵션을 선택하면 서버에 연결하기 위해 SSH를 사용하도록 클라이언트가 구성되고 다른 연결 방법은 구성되지 않습니다.
11. **부트스트랩**을 클릭하여 등록을 시작합니다.

부트스트랩 프로세스가 완료되면 클라이언트가 **시스템** > **시스템 목록**에 나열됩니다.



- SSH 개인 키는 부트스트래핑 프로세스가 진행되는 중에만 저장되며, 부트스트래핑이 완료되자마자 Uyuni 서버에서 삭제됩니다.



- Uyuni(를) 사용해 클라이언트에 새 패키지 또는 업데이트를 설치하면 최종 사용자 라이선스 계약(EULA)이 자동으로 수락됩니다. 패키지 EULA를 검토하려면 Web UI에서 패키지 상세 정보 페이지를 여십시오.

4.1.1.1. 로컬 할당 리포지토리 처리

Uyuni에서 제공하지 않는 클라이언트에 리포지토리를 직접 할당하는 것은 일반적인 사용 사례가 아닙니다. 이로 인해 문제가 발생할 수 있습니다. 그러므로 Salt를 통한 부트스트래핑은 부트스트랩 프로세스를 시작할 때 모든 로컬 리포지토리를 비활성화합니다.

나중에 Highstate 또는 패키지 설치를 실행할 때처럼 채널 상태를 사용할 때마다 로컬에 할당된 모든 리포지토리가 다시 비활성화됩니다.

클라이언트에서 사용되는 모든 소프트웨어 패키지는 Uyuni에서 제공하는 채널에서 가져와야 합니다. 사용자 정의

채널을 생성하는 방법에 대한 자세한 내용은 **Administration > Custom-channels**에서 사용자 정의 채널을 참조하십시오.

4.1.2. 부트스트랩 스크립트로 클라이언트 등록

부트스트랩 스크립트로 클라이언트를 등록하면 파라미터를 제어할 수 있고, 다수의 클라이언트를 한 번에 등록해야 하는 경우 도움이 됩니다. 이 방법은 Salt 및 기존 클라이언트 둘 다에 사용할 수 있습니다.

부트스트랩 스크립트를 사용해 클라이언트를 등록하려면 시작할 템플릿 부트스트랩 스크립트를 생성하는 것이 좋습니다. 이 스크립트는 이후 복사 및 수정할 수 있습니다. 사용자가 실행하는 부트스트랩 스크립트는 클라이언트 등록 시 클라이언트에서 실행되며 필요한 모든 패키지가 클라이언트에 배포되도록 보장합니다. 부트스트랩 스크립트에는 몇 가지 파라미터가 포함되어 있는데, 이 파라미터들은 클라이언트 시스템이 활성화 키 및 GPG 키를 포함한 기본 채널에 할당되도록 보장합니다.

리포지토리 정보를 주의 깊게 확인하여 기본 채널 리포지토리와 일치하게 하는 것이 중요합니다. 리포지토리 정보가 정확히 일치하지 않으면 부트스트랩 스크립트가 패키지를 정확히 다운로드할 수 없습니다.



All clients need a bootstrap repository. It is automatically created and regenerated on the SUSE Manager Server when products are synchronized. A bootstrap repository includes packages for installing Salt on clients, and for registering Salt or traditional clients.

For more information about creating a bootstrap repository, see **Client-configuration > Bootstrap-repository**.

GPG 키 및 Uyuni 클라이언트 도구



Uyuni 클라이언트 도구가 사용하는 GPG 키는 기본적으로 신뢰할 수 없습니다. 부트스트랩 스크립트를 생성할 때 **ORG_GPG_KEY** 파라미터와 함께 공용 키 지문이 포함된 파일의 경로를 추가하십시오.



openSUSE Leap 15 및 SLES 15는 기본적으로 Python 3를 사용합니다. Python 2에 기반을 둔 부트스트랩 스크립트는 openSUSE Leap 15 및 SLE 15 시스템에 대해 다시 생성해야 합니다. Python 2를 사용해 openSUSE Leap 15 또는 SLE 15 시스템을 등록하는 경우 부트스트랩 스크립트는 실패합니다.

4.1.2.1. `mgr-bootstrap` 명령을 사용한 부트스트랩 스크립트 생성

mgr-bootstrap 명령은 사용자 정의 부트스트랩 스크립트를 생성합니다. 초기 등록 및 구성을 단순화하기 위해 Uyuni 클라이언트 시스템은 부트스트랩 스크립트를 사용합니다.

--activation-keys 및 **--script** 인수가 유일한 필수 인수입니다. Uyuni 서버에서 명령줄의 루트로 필수 인수를 사용하여 실행하십시오. 다음과 같이 **<ACTIVATION_KEYS>** 및 **<EDITED_NAME>**을 사용자의 값으로 바꿉니다.

```
mgr-bootstrap --activation-key=<ACTIVATION_KEYS> --script=bootstrap
-<EDITED NAME>.sh
```

mgr-bootstrap 명령은 특정 호스트 이름을 설정하고 특정 GPG 키를 설정하며 등록 방법(기존, salt-minion, salt-bundle)을 설정하는 기능 등 몇 가지 다른 옵션을 제공합니다.

자세한 내용은 **mgr-bootstrap** 사용자 지정 페이지를 참조하거나 **mgr-bootstrap --help** 를 실행하십시오.

4.1.2.2. Web UI에서 부트스트랩 스크립트 생성

Uyuni Web UI를 사용하면 편집 가능한 부트스트랩 스크립트를 생성할 수 있습니다.

절차: 부트스트랩 스크립트 생성

1. Uyuni Web UI에서 **관리** > **관리자 구성** > **부트스트랩 스크립트**로 이동합니다.
2. 기존 클라이언트를 설치하는 경우 **SUSE Manager 구성 - 부트스트랩** 대화 상자에서 **Salt를 사용한 부트스트랩** 확인란의 선택을 취소합니다. Salt 클라이언트의 경우 선택한 상태로 됩니다.
3. 필수 필드는 이전의 설치 단계에서 얻은 값으로 미리 채워져 있습니다. 각 설정에 대한 상세 정보는 **Reference** > **Admin**을 참조하십시오.
4. **업데이트**를 클릭하여 스크립트를 생성합니다.
5. 부트스트랩 스크립트는 서버의 **/srv/www/htdocs/pub/bootstrap** 디렉토리에 생성되고 저장됩니다. 또는 HTTPS를 통해 부트스트랩 스크립트에 액세스할 수 있습니다. **<example.com>** 을 다음과 같은 Uyuni 서버의 호스트 이름으로 바꿉니다.

```
https://<example.com>/pub/bootstrap/bootstrap.sh
```



부트스트랩 스크립트에서 SSL을 비활성화하지 마십시오. Web UI에서 **SSL 활성화** 확인란이 선택되어 있는지, 또는 **USING_SSL=1**이라는 설정이 부트스트랩 스크립트에 있는지 확인합니다. SSL을 비활성화하면 등록 프로세스에 사용자 정의 SSL 인증서가 필요합니다.

For more information about custom certificates, see **Administration** > **Ssl-certs**.

4.1.2.3. 부트스트랩 스크립트 편집

생성한 템플릿 부트스트랩 스크립트를 복사 및 수정하여 사용자 정의할 수 있습니다. Uyuni에서 사용할 부트스트랩 스크립트를 수정할 때 충족해야 할 최소 요구사항은 활성화 키를 포함해야 한다는 것입니다. 대부분의 패키지는 GPG로 서명되므로 패키지를 설치할 시스템에 신뢰할 수 있는 GPG 키도 있어야 합니다.

이 절차에서 활성화 키의 정확한 이름을 알아야 합니다. **홈** > **개요**로 이동하여 **작업** 상자에서 **활성화 키 관리**를 클릭합니다. 채널에 대해 생성된 모든 키는 이 페이지에 나열되어 있습니다. 부트스트랩 스크립트에서 사용하려는 키의 전체 이름을 키 필드에 표시된 대로 정확히 입력해야 합니다. 활성화 키에 대한 자세한 내용은 **Client-configuration** > **Activation-keys**를 참조하십시오.

절차: 부트스트랩 스크립트 수정

1. Uyuni 서버에서 명령줄의 루트 권한으로 다음과 같이 부트스트랩 디렉토리로 변경합니다.

```
cd /srv/www/htdocs/pub/bootstrap/
```

2. 각 클라이언트에서 사용할 템플릿 부트스트랩 스크립트 사본 2개를 생성하여 이름을 변경합니다.

```
cp bootstrap.sh bootstrap-sles12.sh
cp bootstrap.sh bootstrap-sles15.sh
```

3. 수정을 위해 **bootstrap-sles15.sh**를 엽니다. 아래와 같은 텍스트가 나올 때까지 아래로 스크롤합니다. **exit 1**이 파일에 있는 경우 해시 또는 파운드 기호(#)를 줄 첫머리에 입력하여 코멘트 아웃합니다. 이렇게 하면 스크립트가 활성화됩니다. 이 스크립트의 키 이름을 **ACTIVATION_KEYS=** 필드에 다음과 같이 입력합니다.

```
echo "Enable this script: comment (with #'s) this block (or, at least
just"
echo "the exit below)"
echo
#exit 1

# 수정할 수 있으나 아마 정확할 것입니다(초기 설치 중에 생성하지 않은 경우):
# 참고: ACTIVATION_KEYS는 클라이언트 시스템을 부트스트래핑하는 데 *반드시* 사용해야
합니다.
ACTIVATION_KEYS=1-sles15
ORG_GPG_KEY=
```

4. 완료한 후 파일을 저장하고 두 번째 부트스트랩 스크립트에 대해 이 절차를 반복하십시오.



기본적으로 부트스트랩 스크립트는 부트스트랩 리포지토리에서 사용할 수 있는 경우 Salt 클라이언트용 **venv-salt-minion**를 설치하고 부트스트랩 리포지토리에 Salt 번들이 없으면 **salt-minion**을 설치하려고 시도합니다. 어떤 이유로 필요한 경우 Salt 번들을 설치하지 않고 **salt-minion**을 계속 사용하는 것이 가능합니다.

자세한 내용은 **Client-configuration > Contact-methods-saltbundle**에서 확인할 수 있습니다.

4.1.2.4. 클라이언트 연결

스크립트 생성을 완료했으면 이 스크립트를 사용해 클라이언트를 등록할 수 있습니다.

절차: 부트스트랩 스크립트 실행

1. Uyuni 서버에서 루트 권한으로 로그인합니다. 명령 프롬프트에서 다음과 같이 부트스트랩 디렉토리로 변경합니다.

```
cd /srv/www/htdocs/pub/bootstrap/
```

2. 다음 명령을 실행하여 클라이언트에서 부트스트랩 스크립트를 실행하고, **EXAMPLE.COM**을 클라이언트의 호스트 이름으로 교체합니다.

```
cat bootstrap-sles15.sh | ssh root@EXAMPLE.COM /bin/bash
```

3. 또는 클라이언트에서 다음 명령을 실행합니다.

```
curl -Sks https://server_hostname/pub/bootstrap/bootstrap-sles15.sh | /bin/bash
```



문제를 해결하려면 **bash** 명령을 사용하여 부트스트랩 스크립트를 실행해 확인하십시오.

이 스크립트는 앞서 생성한 리포지토리 디렉토리에 있는 필수 종속성을 다운로드합니다.

- 스크립트가 실행되고 나면 Uyuni Web UI를 열고 **시스템 > 개요**로 이동하여 새 클라이언트가 나열되어 있는지 확인하여 클라이언트가 올바르게 등록되었는지 확인할 수 있습니다.
- 스크립트를 사용하여 Salt 클라이언트를 등록한 경우 Uyuni Web UI를 열고 **Salt > 키**로 이동하여 클라이언트 키를 수락합니다.



Uyuni를 사용해 클라이언트에 새 패키지 또는 업데이트를 설치하면 최종 사용자 라이선스 계약(EULA)이 자동으로 수락됩니다. 패키지 EULA를 검토하려면 Web UI에서 패키지 상세 정보 페이지를 여십시오.

4.1.3. 명령줄에서 등록(Salt)

4.1.3.1. 수동 Salt 클라이언트 등록

대부분의 경우 Salt 클라이언트는 기본 부트스트랩 방법으로 정확히 등록됩니다. 하지만 Salt를 사용하면 클라이언트의 Salt Minion 파일을 편집하고 서버의 전체 도메인 이름(FQDN)을 제공하여 수동으로 Uyuni 서버에 클라이언트를 등록할 수 있습니다. 이 방법에서는 서버에 대한 4505 및 4506 인바운드 포트가 사용됩니다. 이 방법은 해당 포트가 열려 있는지 확인하는 것 외에 Uyuni 서버에서 구성할 필요가 없습니다.



기존 클라이언트의 경우 명령줄에서 등록하는 것도 가능하지만 그려려면 더 많은 단계를 거쳐야 합니다. 여기서는 이에 관해 다루지 않겠습니다. 부트스트랩 스크립트 절차를 사용해 기존 클라이언트를 등록하십시오. 자세한 내용은 [registration-bootstrap.pdf](#)를 참조하십시오.

이 절차를 수행하려면 Salt 클라이언트에 **venv-salt-minion** (Salt 번들) 또는 **salt-minion** 패키지를 설치한 후 등록해야 합니다. 둘 다 다른 위치에서 구성 파일을 사용하며 파일 이름은 동일하게 유지됩니다. systemd 서비스 파일 이름이 다릅니다.



이 방법으로 부트스트랩하면 클라이언트 도구 채널 또는 공식 SUSE 배포판의 일부인 **salt-minion**을 사용하는 경우에만 작동합니다.

4.1.3.1.1. Salt Bundle 구성

Salt Bundle(**venv-salt-minion**)

- /etc/venv-salt-minion/**
- /etc/venv-salt-minion/minion**
- /etc/venv-salt-minion/minion.d/NAME.conf**

- systemd 서비스 파일: **venv-salt-minion.service**

Salt 번들에 대한 자세한 내용은 **Client-configuration > Contact-methods-saltbundle**에서 참조하십시오.

절차: Salt Bundle 구성 파일로 클라이언트 등록

1. Salt 클라이언트에서 **minion** 구성 파일을 엽니다. 구성 파일의 위치는 다음 중 하나입니다.

```
/etc/venv-salt-minion/minion
```

또는

```
/etc/venv-salt-minion/minion.d/NAME.conf
```

2. 파일에서 Uyuni 서버 또는 프록시의 FQDN과 활성화 키(있는 경우)를 추가하거나 편집합니다. 또한 아래 나열된 다른 구성 파라미터를 추가합니다.

```
master: SERVER.EXAMPLE.COM

세분화:
susemanager:
activation_key: "<Activation_Key_Name>"

server_id_use_crc: adler32
enable_legacy_startup_events: False
enable_fqdns_grains: False
```

3. 다음과 같이 **venv-salt-minion** 서비스를 재시작합니다.

```
systemctl restart venv-salt-minion
```

4. Uyuni 서버에서 새로운 클라이언트 키를 수락하고, 다음과 같이 **<client>**를 클라이언트의 이름으로 교체합니다.

```
salt-key -a '<client>'
```

4.1.3.1.2. Salt Minion 구성

Salt Minion(**salt-minion**)

- **/etc/salt/**
- **/etc/salt/minion**
- **/etc/salt/minion.d/NAME.conf**

- systemd 서비스 파일: **salt-minion.service**

절차: Salt Minion 구성 파일로 클라이언트 등록

1. Salt 클라이언트에서 **minion** 구성 파일을 엽니다. 구성 파일의 위치는 다음 중 하나입니다.

```
/etc/salt/minion
```

또는

```
/etc/salt/minion.d/NAME.conf
```

2. 파일에서 Uyuni 서버 또는 프록시의 FQDN과 활성화 키(있는 경우)를 추가하거나 편집합니다. 또한 아래 나열된 다른 구성 파라미터를 추가합니다.

```
master: SERVER.EXAMPLE.COM
```

세분화:

```
susemanager:
activation_key: "<Activation_Key_Name>"
```

```
server_id_use_crc: adler32
enable_legacy_startup_events: False
enable_fqdns_grains: False
```

3. 다음과 같이 **salt-minion** 서비스를 재시작합니다.

```
systemctl restart salt-minion
```

4. Uyuni 서버에서 새로운 클라이언트 키를 수락하고, 다음과 같이 <client></i>를 클라이언트의 이름으로 교체합니다.

```
salt-key -a '<client>'
```

Salt 미니언 구성 파일에 대한 자세한 내용은 <https://docs.saltstack.com/en/latest/ref/configuration/minion.html>을 참조하십시오.

4.2. SUSE 클라이언트 등록

You can register SUSE Linux Enterprise clients to your Uyuni Server.

The method and details varies depending on the operating system of the client.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.



- Do not register a Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

4.2.1. SUSE Linux Enterprise 클라이언트 등록

이 섹션에는 다음과 같은 SUSE Linux Enterprise 운영 체제를 실행하는 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

- SUSE Linux Enterprise Server 15 SP1
- SUSE Linux Enterprise Server 15 SP2
- SUSE Linux Enterprise Server 15 SP3
- SUSE Linux Enterprise Server 15 SP4
- SUSE Linux Enterprise Server 15 SP5

다음을 포함하여 모든 SUSE Linux Enterprise 제품을 준비하려면 이 장의 지침을 사용하십시오.

- SUSE Linux Enterprise Server for SAP
- SUSE Linux Enterprise 데스크톱
- SUSE Linux Enterprise
- SUSE Linux Enterprise Real Time

이전 SUSE Linux Enterprise 버전 및 서비스 팩에서도 이러한 지침을 사용할 수 있습니다.

4.2.1.1. 소프트웨어 채널 추가



- 다음 섹션에서 설명은 종종 **x86_64** 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

SUSE Linux Enterprise 클라이언트를 Uyuni 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

이 절차에 필요한 제품은 다음과 같습니다.

표 16. SLE 제품 - WebUI

OS Version	Product Name
SUSE Linux Enterprise Server 12 SP5	SUSE Linux Enterprise Server 12 SP5 x86_64
SUSE Linux Enterprise Server 15 SP1	SUSE Linux Enterprise Server 15 SP1 x86_64
SUSE Linux Enterprise Server 15 SP2	SUSE Linux Enterprise Server 15 SP2 x86_64
SUSE Linux Enterprise Server 15 SP3	SUSE Linux Enterprise Server 15 SP3 x86_64
SUSE Linux Enterprise Server 15 SP4	SUSE Linux Enterprise Server 15 SP4 x86_64

OS Version	Product Name
SUSE Linux Enterprise Server 15 SP5	SUSE Linux Enterprise Server 15 SP5 x86_64

절차: 소프트웨어 채널 추가

1. Uyuni Web UI에서 **관리 > 설치 마법사 > 제품**으로 이동합니다.
2. 검색 창을 사용하여 클라이언트 운영 체제 및 아키텍처에 적합한 제품을 찾고 해당 제품을 확인하십시오. 모든 필수 채널은 자동으로 확인됩니다. 또한, **추천 포함** 토글이 켜져 있으면 모든 추천 채널이 확인됩니다. 관련 제품의 전체 목록을 보려면 화살표를 클릭하고 필요한 추가 제품이 선택되어 있는지 확인하십시오.
3. **제품 추가**를 클릭하고 제품이 동기화를 마칠 때까지 기다립니다.

또는 명령 프롬프트에서 채널을 추가할 수 있습니다. 이 절차에 필요한 채널은 다음과 같습니다.

표 17. SLE 제품 - CLI

OS Version	Base Channel
SUSE Linux Enterprise Server 12 SP5	sle-product-sles12-sp5-pool-x86_64
SUSE Linux Enterprise Server 15 SP1	sle-product-sles15-sp1-pool-x86_64
SUSE Linux Enterprise Server 15 SP2	sle-product-sles15-sp2-pool-x86_64
SUSE Linux Enterprise Server 15 SP3	sle-product-sles15-sp3-pool-x86_64
SUSE Linux Enterprise Server 15 SP4	sle-product-sles15-sp4-pool-x86_64
SUSE Linux Enterprise Server 15 SP5	sle-product-sles15-sp5-pool-x86_64

이전 제품의 채널 이름을 찾으려면 Uyuni 서버의 명령 프롬프트에서 root 권한으로 **mgr-sync** 명령을 사용합니다.

```
mgr-sync list --help
```

그런 다음 원하는 인수를 지정합니다. 예를 들어, **channels**:

```
mgr-sync list channels [-c]
```

명령 프롬프트에서 소프트웨어 채널 추가

1. Uyuni 서버의 명령 프롬프트에서 root 권한으로 다음과 같이 **mgr-sync** 명령을 사용해 적절한 채널을 추가합니다.

```
mgr-sync add channel <channel_label_1>
mgr-sync add channel <channel_label_2>
mgr-sync add channel <channel_label_n>
```

2. 동기화가 자동으로 시작됩니다. 채널을 수동으로 동기화하려면 다음 명령을 사용하십시오.

```
mgr-sync sync --with-children <channel_name>
```

- 계속하기 전에 동기화가 완료되었는지 확인합니다.

클라이언트 도구를 추가하려면 명령 프롬프트에서 다음 채널을 추가하십시오.

표 18. SUSE Linux Enterprise 채널 - CLI

OS Version	Client Channel
SUSE Linux Enterprise Server 12 SP5	sles12-sp5-uyuni-client
SUSE Linux Enterprise Server 15 SP1	sles15-sp1-uyuni-client
SUSE Linux Enterprise Server 15 SP2	sles15-sp2-uyuni-client
SUSE Linux Enterprise Server 15 SP3	sles15-sp3-uyuni-client
SUSE Linux Enterprise Server 15 SP4	sles15-sp4-uyuni-client
SUSE Linux Enterprise Server 15 SP5	sles15-sp5-uyuni-client

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root로 **spacewalk-common-channels** 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

- 계속하기 전에 동기화가 완료되었는지 확인합니다.

4.2.1.2. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

- Uyuni Web UI에서 **소프트웨어**, **관리**, **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
- 리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

- Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.



- SUSE Linux Enterprise 채널은 규모가 매우 클 수 있습니다. 때로 동기화에 몇 시간이 걸릴 수 있습니다.

4.2.1.3. GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.



- Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

GPG 키에 대한 자세한 내용은 [Client-configuration > Gpg-keys](#)에서 확인할 수 있습니다.



- SUSE Linux Enterprise Server 15 및 SUSE Linux Enterprise Server 12 클라이언트에 동일한 GPG 키를 사용합니다. 올바른 키는 `sle12-gpg-pubkey-39db7c82.key`입니다.

4.2.1.4. 클라이언트 등록

SUSE Linux Enterprise 클라이언트를 등록하려면 부트스트랩 리포지토리가 필요합니다. 기본적으로 부트스트랩 리포지토리는 자동으로 생성되며 동기화된 모든 제품에 대해 매일 재생성됩니다. 명령 프롬프트에서 다음 명령을 사용해 부트스트랩 리포지토리를 수동으로 생성할 수 있습니다.

```
mgr-create-bootstrap-repo
```

클라이언트 등록에 대한 자세한 내용은 [Client-configuration > Registration-overview](#)에서 참조하십시오.

4.2.2. SLE Micro 클라이언트 등록

이 섹션에는 다음과 같은 SLE Micro 운영 체제를 실행하는 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

- SLE Micro 5.1, 5.2, and 5.3 x86-64
- SLE Micro 5.1, 5.2, and 5.3 ARM64
- SLE Micro 5.1, 5.2, and 5.3 IBM Z (s390x)



- SLE Micro 클라이언트에 대한 지원은 테스트 목적을 위해 기술 미리보기로 제공되며 이 단계에서는 모든 기능이 완전히 작동하지 않습니다. 이 기능은 Uyuni 이후 버전에서 완전하게 지원될 예정입니다. 프로덕션 시스템에서 이 기능을 사용하지 마십시오.

SLE Micro는 에지 컴퓨팅을 위해 특별히 제작된 매우 안정적인 경량 운영 체제입니다. 그리고 SUSE Linux Enterprise의 엔터프라이즈 강화 보안 및 규정 준수 구성 요소를 활용하며, 최신의 변경 불가능하고 개발자 친화적인 OS 플랫폼과 병합합니다.

SLE Micro는 트랜잭션 업데이트를 사용합니다. 트랜잭션 업데이트는 원자적(모든 업데이트가 성공한 경우에만 모든 업데이트가 적용됨)이며 룰백을 지원합니다. 시스템이 재부팅될 때까지 변경 사항이 적용되지 않으므로 실행 중인 시스템에 영향을 주지 않습니다. 이와 관련된 정보는 **Systems > 세부 정보 > 개요** 하위 탭에 표시됩니다.

트랜잭션 업데이트 및 재부팅에 대한 자세한 내용은 <https://documentation.suse.com/sles/html/SLES-all/cha-transactional-updates.html>에서 확인할 수 있습니다.

4.2.2.1. 소프트웨어 채널 추가

SLE Micro 클라이언트를 Uyuni 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.



- 다음 섹션에서 설명은 종종 **x86_64** 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

이 절차에 필요한 제품은 다음과 같습니다.

표 19. SLE Micro 제품 - WebUI

OS Version	Product Name
SLE Micro 5.1 x86-64	SUSE Linux Enterprise Micro 5.1 x86_64
SLE Micro 5.1 ARM64	SUSE Linux Enterprise Micro 5.1 aarch64
SLE Micro 5.1 s390x	SUSE Linux Enterprise Micro 5.1 s390x
SLE Micro 5.2 x86-64	SUSE Linux Enterprise Micro 5.2 x86_64
SLE Micro 5.2 ARM64	SUSE Linux Enterprise Micro 5.2 aarch64
SLE Micro 5.2 s390x	SUSE Linux Enterprise Micro 5.2 s390x
SLE Micro 5.3 x86-64	SUSE Linux Enterprise Micro 5.3 x86_64
SLE Micro 5.3 ARM64	SUSE Linux Enterprise Micro 5.3 aarch64
SLE Micro 5.3 s390x	SUSE Linux Enterprise Micro 5.3 s390x

절차: 소프트웨어 채널 추가

1. Uyuni Web UI에서 **관리 > 설치 마법사 > 제품**으로 이동합니다.
2. 검색 창을 사용하여 클라이언트 운영 체제 및 아키텍처에 적합한 제품을 찾고 해당 제품을 확인하십시오. 모든 필수 채널은 자동으로 확인됩니다. 또한, **추천 포함** 토글이 켜져 있으면 모든 추천 채널이 확인됩니다. 관련 제품의 전체 목록을 보려면 화살표를 클릭하고 필요한 추가 제품이 선택되어 있는지 확인하십시오.
3. **제품 추가**를 클릭하고 제품이 동기화를 마칠 때까지 기다립니다.

또는 명령 프롬프트에서 채널을 추가할 수 있습니다. 이 절차에 필요한 채널은 다음과 같습니다.

표 20. SLE Micro 제품 - CLI

OS Version	Base Channel	Updates Channel
SLE Micro 5.1 x86_64	suse-microos-5.1-pool-x86_64	suse-microos-5.1-updates-x86_64
SLE Micro 5.1 ARM64	suse-microos-5.1-pool-aarch64	suse-microos-5.1-updates-aarch64
SLE Micro 5.1 IBM Z (s390x)	suse-microos-5.1-pool-s390x	suse-microos-5.1-updates-s390x
SLE Micro 5.2 x86_64	suse-microos-5.2-pool-x86_64	suse-microos-5.2-updates-x86_64
SLE Micro 5.2 ARM64	suse-microos-5.2-pool-aarch64	suse-microos-5.2-updates-aarch64
SLE Micro 5.2 IBM Z (s390x)	suse-microos-5.2-pool-s390x	suse-microos-5.2-updates-s390x
SLE Micro 5.3 x86_64	sle-micro-5.3-pool-x86_64	sle-micro-5.3-updates-x86_64
SLE Micro 5.3 ARM64	sle-micro-5.3-pool-arm64	sle-micro-5.3-updates-arm64
SLE Micro 5.3 IBM Z (s390x)	sle-micro-5.3-pool-s390x	sle-micro-5.3-updates-s390x

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root 권한으로 다음과 같이 **mgr-sync** 명령을 사용해 적절한 채널을 추가합니다.

```
mgr-sync add channel <channel_label_1>
mgr-sync add channel <channel_label_2>
mgr-sync add channel <channel_label_n>
```

- 동기화가 자동으로 시작됩니다. 채널을 수동으로 동기화하려면 다음 명령을 사용하십시오.

```
mgr-sync sync --with-children <channel_name>
```

- 계속하기 전에 동기화가 완료되었는지 확인합니다.

클라이언트 도구를 추가하려면 명령 프롬프트에서 다음 채널을 추가하십시오.

표 21. SLE Micro 채널 - CLI

OS Version	Client Channel
SLE Micro 5.1	suse-microos-5.1-uyuni-client
SLE Micro 5.2	suse-microos-5.2-uyuni-client
SLE Micro 5.3	sle-micro-5.3-uyuni-client

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root로 **spacewalk-common-channels** 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. **automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 계속하기 전에 동기화가 완료되었는지 확인합니다.

4.2.2.2. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

1. Uyuni Web UI에서 **소프트웨어**, **관리**, **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
2. **리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

4.2.2.3. 클라이언트 등록



SLE Micro clients require reboot after registering. Reboot is automatically scheduled after registration is completed, but it is respecting the default reboot manager maintenance window. This window may be several hours after the client is registered. To speed up SLE Micro registration, manually reboot the client after the registration script finishes.

To register your SLE Micro clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

```
mgr-create-bootstrap-repo
```

클라이언트 등록에 대한 자세한 내용은 **Client-configuration** > **Registration-overview**에서 참조하십시오.

4.3. openSUSE 클라이언트 등록

You can register openSUSE and openSUSE Leap Micro clients to your Uyuni Server. The method and details varies depending on the operating system of the client.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.



Do not register a Uyuni Server to itself. The Uyuni Server must be managed individually or by using another separate Uyuni Server. For more information about using multiple servers, see [Specialized-guides > Large-deployments](#).

4.3.1. openSUSE Leap 클라이언트 등록

이 섹션에는 openSUSE 운영 체제를 실행하는 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다. Uyuni는 Salt를 사용하는 openSUSE Leap 15 클라이언트를 지원합니다. 기존 클라이언트는 지원하지 않습니다.

부트스트래핑은 openSUSE 클라이언트 시작과 초기 상태 실행 수행(예: 리포지토리 설정, 프로파일 업데이트 수행)에 대해 지원됩니다.

4.3.1.1. 소프트웨어 채널 추가

openSUSE 클라이언트를 Uyuni 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

현재 지원되는 아키텍처는 **x86_64** 및 **aarch64**입니다. 지원되는 제품 및 아키텍처의 전체 목록은 [Client-configuration > Supported-features](#)에서 참조하십시오.



다음 섹션에서 설명은 종종 **x86_64** 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

예를 들어, **x86_64** 아키텍처에서 작업 시 필요한 제품은 다음과 같습니다.

표 22. OpenSUSE 채널 - CLI

OS 버전	기본 채널	클라이언트 채널	업데이트 채널	비-OSS 채널	비-OSS 업데이트 채널
openSUSE Leap 15.1	opensuse_leap 15_1	opensuse_leap 15_1-uyuni-client	opensuse_leap 15_1-updates	opensuse_leap 15_1-non-oss	opensuse_leap 15_1-non-oss-updates
openSUSE Leap 15.2	opensuse_leap 15_2	opensuse_leap 15_2-uyuni-client	opensuse_leap 15_2-updates	opensuse_leap 15_2-non-oss	opensuse_leap 15_2-non-oss-updates

표 23. OpenSUSE 채널 - CLI

OS Version	Base Channel	Client Channel	Updates Channel	Non-OSS Channel	Non-OSS Updates Channel	Backports Updates Channel	SLE Updates Channel
openSUSE Leap 15.3	opensuse_l eap15_3	opensuse_l eap15_3-uyuni-client	opensuse_l eap15_3-updates	opensuse_l eap15_3-non-oss	opensuse_l eap15_3-non-oss-updates	opensuse_l eap15_3-backports-updates	opensuse_l eap15_3-sle-updates
openSUSE Leap 15.4	opensuse_l eap15_4	opensuse_l eap15_4-uyuni-client	opensuse_l eap15_4-updates	opensuse_l eap15_4-non-oss	opensuse_l eap15_4-non-oss-updates	opensuse_l eap15_4-backports-updates	opensuse_l eap15_4-sle-updates
openSUSE Leap 15.5	opensuse_l eap15_5	opensuse_l eap15_5-uyuni-client	opensuse_l eap15_5-updates	opensuse_l eap15_5-non-oss	opensuse_l eap15_5-non-oss-updates	opensuse_l eap15_5-backports-updates	opensuse_l eap15_5-sle-updates

명령 프롬프트에서 소프트웨어 채널 추가

1. Uyuni 서버의 명령 프롬프트에서 root로 **spacewalk-common-channels** 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. **automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 계속하기 전에 동기화가 완료되었는지 확인합니다.

4.3.1.2. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

1. Uyuni Web UI에서 **소프트웨어**, **관리**, **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
2. **리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.



openSUSE 채널은 규모가 매우 클 수 있습니다. 때로 동기화에 몇 시간이 걸릴 수 있습니다.

4.3.1.3. GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.



Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

GPG 키에 대한 자세한 내용은 [Client-configuration > Gpg-keys](#)에서 확인할 수 있습니다.

4.3.1.4. 클라이언트 등록

openSUSE 클라이언트를 등록하려면 부트스트랩 리포지토리가 필요합니다. 기본적으로 부트스트랩 리포지토리는 자동으로 생성되며 동기화된 모든 제품에 대해 매일 재생성됩니다. 명령 프롬프트에서 다음 명령을 사용해 부트스트랩 리포지토리를 수동으로 생성할 수 있습니다.

```
mgr-create-bootstrap-repo
```

클라이언트 등록에 대한 자세한 내용은 [Client-configuration > Registration-overview](#)에서 참조하십시오.

4.3.2. Registering openSUSE Leap Micro Clients

This section contains information about registering clients running these openSUSE Leap Micro operating systems:

- openSUSE Leap Micro 5.3 x86-64
- openSUSE Leap Micro 5.3 ARM64

The openSUSE Leap Micro is an ultra-reliable, lightweight operating system purpose built for edge computing. It leverages the enterprise hardened security and compliance components of SUSE Linux Enterprise and merges them with a modern, immutable, developer-friendly OS platform.

The openSUSE Leap Micro uses transactional updates. Transactional updates are atomic (all updates are applied only if all updates succeed) and support rollbacks. They do not affect the running system because no changes are activated until the system is rebooted. This information is displayed in the [Systems > Details > Overview](#) subtab.

트랜잭션 업데이트 및 재부팅에 대한 자세한 내용은 <https://documentation.suse.com/sles/html/SLES-all/cha-transactional-updates.html>에서 확인할 수 있습니다.

표 24. OpenSUSE 채널 - CLI

OS Version	Base Channel	Client Channel	SLE Updates Channel
openSUSE Leap Micro 5.3	opensuse_micro5_3	opensuse_micro5_3-sle-updates	opensuse_micro5_3-uyuni-client

명령 프롬프트에서 소프트웨어 채널 추가

1. Uyuni 서버의 명령 프롬프트에서 root로 **spacewalk-common-channels** 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. **automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 계속하기 전에 동기화가 완료되었는지 확인합니다.

4.3.2.1. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

1. Uyuni Web UI에서 **소프트웨어**, **관리**, **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
2. **리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.



openSUSE Leap Micro channels can be very large. Synchronization can sometimes take several hours.

4.3.2.2. 클라이언트 등록



openSUSE Leap Micro clients require reboot after registering. Reboot is automatically scheduled after registration is completed, but it is respecting the default reboot manager maintenance window. This window may be several hours after the client is registered. To speed up openSUSE Leap Micro

- registration, manually reboot the client after the registration script finishes.

To register openSUSE Leap Micro clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using the command:

```
mgr-create-bootstrap-repo
```

For more information on registering clients, see [Client-configuration > Registration-overview](#).

4.4. Alibaba Cloud Linux 클라이언트 등록

Alibaba Cloud Linux 클라이언트를 Uyuni 서버에 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 다릅니다.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.

4.4.1. Alibaba Cloud Linux 클라이언트 등록

이 섹션에는 Alibaba Cloud Linux 운영 체제를 실행하는 기존 및 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어.

기존 스택은 Alibaba Cloud Linux 2에서 사용할 수 있지만 지원되지는 않습니다. Alibaba Cloud Linux 2 클라이언트는 Salt 클라이언트로만 지원됩니다.



- 일부 Alibaba Cloud Linux 2 인스턴스를 등록하려면 2회 시도해야 합니다.

4.4.1.1. 소프트웨어 채널 추가

Alibaba Cloud Linux 클라이언트를 Uyuni 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.



- 다음 섹션에서 설명은 종종 **x86_64** 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

이 절차에 필요한 채널은 다음과 같습니다.

표 25. Alibaba Cloud Linux 채널 - CLI

OS 버전	Core Channel	Updates Channel	Client Channel
Alibaba Cloud Linux 2	alibaba-2	alibaba-2-updates	alibaba-2-uyuni-client

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root로 **spacewalk-common-channels** 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. **automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 계속하기 전에 동기화가 완료되었는지 확인합니다.



spacewalk-common-channels 가 제공하는 클라이언트 도구 채널은 SUSE가 아닌 Uyuni가 공급합니다.

4.4.1.2. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

1. Uyuni Web UI에서 **소프트웨어**, **관리**, **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
2. **리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

4.4.1.3. 활성화 키 생성

Alibaba Cloud Linux 채널에 연결된 활성화 키를 생성해야 합니다.

활성화 키에 대한 자세한 내용은 **Client-configuration** > **Activation-keys**를 참조하십시오.

4.4.1.4. 클라이언트 등록

Alibaba Cloud Linux 클라이언트는 기타 모든 클라이언트와 동일한 방식으로 등록됩니다.

일부 Alibaba Cloud Linux 2 인스턴스는 첫 번째 시도에서 등록에 실패합니다.

이는 Alibaba Cloud Linux 2 이미지의 알 수 없는 버그로 인한 것입니다.

python-urlgrabber3 패키지는 Python pip 패키지 및 RPM 패키지로 제공되며, 이로 인해 첫 번째 등록 시도에서

충돌이 발생할 수 있습니다.

인스턴스가 영향을 받는 이미지 버전 중 하나에 해당하는 경우 두 번째 등록 시도에서 클라이언트가 올바르게 등록되어야 합니다.

클라이언트 등록에 대한 자세한 설명은 [Client-configuration > Registration-overview](#)을 참조하십시오.

4.5. AlmaLinux 클라이언트 등록

AlmaLinux 클라이언트를 Uyuni 서버에 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 다릅니다.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.

4.5.1. AlmaLinux 클라이언트 등록

이 섹션에는 AlmaLinux 운영 체제를 실행하는 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

기존 클라이언트는 AlmaLinux에서 사용할 수 없습니다. AlmaLinux 클라이언트는 Salt 클라이언트로만 지원됩니다.



AWS에서 생성할 때 AlmaLinux 인스턴스에서는 항상 `/etc/machine-id`에서 `machine-id` 가 동일합니다. 인스턴스가 생성된 후에 `machine-id`를 다시 생성해야 합니다. 자세한 설명은 [Administration > Troubleshooting](#)에서 확인할 수 있습니다.

4.5.1.1. 소프트웨어 채널 추가



AlmaLinux 클라이언트를 Uyuni에 등록하는 기능은 `목표` 정책으로 `강제` 하는 기본 SELinux 구성으로 테스트됩니다. SELinux를 비활성화해야 AlmaLinux 클라이언트를 Uyuni에 등록할 수 있는 것은 아닙니다.

AlmaLinux 클라이언트를 Uyuni 서버에 등록하려면 먼저 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

현재 지원되는 아키텍처는 `x86_64` 및 `aarch64`이며, 버전 9의 경우 추가적으로 `ppc64le` 및 `s390x`가 지원됩니다. 지원되는 제품 및 아키텍처의 전체 목록은 [Client-configuration > Supported-features](#)에서 확인할 수 있습니다.



다음 섹션에서 설명은 종종 `x86_64` 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

이 절차에 필요한 채널은 다음과 같습니다.

표 26. AlmaLinux 채널 - CLI

OS 버전	기본 채널	클라이언트 채널	AppStream 채널
AlmaLinux 9	almalinux9	almalinux9-uyuni-client	almalinux9-appstream
AlmaLinux 8	almalinux8	almalinux8-uyuni-client	almalinux8-appstream

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root 권한으로 **spacewalk-common-channels** 명령을 사용해 적절한 채널을 추가합니다. 다음과 같이 올바른 아키텍처를 지정했는지 확인합니다.

```
spacewalk-common-channels \
-a <architecture> \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

- automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

- 계속하기 전에 동기화가 완료되었는지 확인합니다.



spacewalk-common-channels 가 제공하는 클라이언트 도구 채널은 SUSE가 아닌 Uyuni가 공급합니다.



AlmaLinux 9 및 AlmaLinux 8 클라이언트의 경우 기본 채널과 AppStream 채널을 모두 추가하십시오. 사용자에게는 두 채널의 패키지가 모두 필요합니다. 두 채널을 모두 추가하지 않으면 패키지가 누락되어 부트스트랩 리포지토리를 생성할 수 없습니다.

모듈형 채널을 사용하는 경우 AlmaLinux 8 클라이언트에서 Python 3.6 모듈 스트림을 활성화해야 합니다. Python 3.6이 제공되지 않으면 **spacecmd** 패키지의 설치가 실패합니다.



AppStream 채널에서 사용할 수 있는 패키지의 수가 업스트림과 Uyuni 채널 간에 서로 약간 불일치함을 알 수 있습니다. 또한 다른 시점에 추가한 동일 채널을 비교하면 그 수가 다른 것을 알 수 있습니다. 이는 AlmaLinux가 리포지토리를 관리하는 방식으로 인한 것입니다. AlmaLinux는 새 버전이 릴리스되면 이전 버전의 패키지를 제거하지만 Uyuni는 사용 기간과 관계없이 모두 그대로 유지합니다.



AppStream 리포지토리는 모듈형 패키지를 제공합니다. 이로 인해 Uyuni Web UI에 부정확한 패키지 정보가 표시됩니다. Web UI 또는 API를 사용해 모듈형 리포지토리에서 직접 설치하거나 업그레이드하는 등의 패키지 작업을 수행할 수 없습니다.

또는 Salt 상태를 사용해 Salt 클라이언트에서 모듈형 패키지를 관리하거나 클라이언트에서 **dnf** 명령을 사용할 수 있습니다. CLM에 대한 자세한 내용은 **Administration > Content-lifecycle**을 참조하십시오.

4.5.1.2. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

- Uyuni Web UI에서 **소프트웨어 > 관리 > 채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.

2. 리포지토리 탭으로 이동한 다음, 동기화를 클릭하고 동기화 상태를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 tail 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

4.5.1.3. 활성화 키 생성

AlmaLinux 채널에 연결된 활성화 키를 생성해야 합니다.

활성화 키에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.

4.5.1.4. GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.



Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

GPG 키에 대한 자세한 내용은 [Client-configuration > Gpg-keys](#)에서 확인할 수 있습니다.

4.5.1.5. 클라이언트 등록

AlmaLinux 클라이언트는 기타 모든 클라이언트와 동일한 방식으로 등록됩니다. 자세한 설명은 [Client-configuration > Registration-overview](#)를 참조하십시오.

4.5.1.6. 정오표 관리

AlmaLinux 클라이언트를 업데이트할 때 패키지에는 업데이트에 대한 메타데이터가 포함됩니다.

4.6. Amazon Linux 클라이언트 등록

Amazon Linux 클라이언트를 Uyuni 서버에 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 다릅니다.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.

4.6.1. Amazon Linux 클라이언트 등록

이 섹션에는 Amazon Linux 운영 체제를 실행하는 기존 및 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

기존 클라이언트는 Amazon Linux 2에서 사용할 수 없습니다. Amazon Linux 2 클라이언트는 Salt 클라이언트로만 지원됩니다.



- AWS에서 생성될 때 Amazon Linux 인스턴스에서는 항상 `/etc/machine-id` 에서 **machine-id** 가 동일합니다. 인스턴스가 생성된 후에 **machine-id** 를 다시 생성해야 합니다. 자세한 내용은 **Administration > Troubleshooting**에서 확인할 수 있습니다.

4.6.1.1. 소프트웨어 채널 추가

Amazon Linux 클라이언트를 Uyuni 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

현재 지원되는 아키텍처는 **x86_64** 및 **aarch64**입니다. 지원되는 제품 및 아키텍처의 전체 목록은 **Client-configuration > Supported-features**에서 참조하십시오.



- 다음 섹션에서 설명은 종종 **x86_64** 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

이 절차에 필요한 채널은 다음과 같습니다.

표 27. Amazon Linux 채널 - CLI

OS 버전	코어 채널	클라이언트 채널
Amazon Linux 2	amazonlinux2-core	amazonlinux2-uyuni-client



- Amazon Linux 인스턴스에서 Docker를 사용하려면 **amazonlinux2-extra-docker** 채널도 추가하고 동기화해야 합니다.

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root로 **spacewalk-common-channels** 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 계속하기 전에 동기화가 완료되었는지 확인합니다.



spacewalk-common-channels 가 제공하는 클라이언트 도구 채널은 SUSE가 아닌 Uyuni가 공급합니다.

4.6.1.2. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

1. Uyuni Web UI에서 **소프트웨어**, **관리**, **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
2. **리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

4.6.1.3. 활성화 키 생성

Amazon Linux 채널에 연결된 활성화 키를 생성해야 합니다.

활성화 키에 대한 자세한 내용은 **Client-configuration > Activation-keys**를 참조하십시오.

4.6.1.4. GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.



Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

GPG 키에 대한 자세한 내용은 **Client-configuration > Gpg-keys**에서 확인할 수 있습니다.

4.6.1.5. 클라이언트 등록

Amazon Linux 클라이언트는 기타 모든 클라이언트와 동일한 방식으로 등록됩니다. 자세한 설명은 **Client-configuration > Registration-overview**를 참조하십시오.

4.7. CentOS 클라이언트 등록

CentOS 클라이언트를 Uyuni 서버에 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 달라집니다.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.

4.7.1. CentOS 클라이언트 등록

이 섹션에는 CentOS 운영 체제를 실행하는 기존 및 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

기존 클라이언트는 CentOS 8에서 사용할 수 없습니다. CentOS 8 클라이언트는 Salt 클라이언트로만 지원됩니다.



You are responsible for arranging access to CentOS base media repositories and CentOS installation media, as well as connecting Uyuni Server to the CentOS content delivery network.



CentOS 클라이언트를 Uyuni에 등록하는 기능은 **목표** 정책으로 **강제** 하는 기본 SELinux 구성으로 테스트됩니다. SELinux를 비활성화해야 CentOS 클라이언트를 Uyuni에 등록할 수 있는 것은 아닙니다.

4.7.1.1. 소프트웨어 채널 추가

CentOS 클라이언트를 Uyuni 서버에 등록하려면 먼저 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

현재 지원되는 아키텍처는 **x86_64** 및 **aarch64**입니다. 지원되는 제품 및 아키텍처의 전체 목록은 [Client-configuration > Supported-features](#)에서 참조하십시오.



다음 섹션에서 설명은 종종 **x86_64** 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

이 절차에 필요한 채널은 다음과 같습니다.

표 28. CentOS 채널 - CLI

OS 버전	기본 채널	클라이언트 채널	업데이트/Appstream 채널
CentOS 7	centos7	centos7-uyuni-client	centos7-updates

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root 권한으로 **spacewalk-common-channels** 명령을 사용해 적절한 채널을 추가합니다. 다음과 같이 올바른 아키텍처를 지정했는지 확인합니다.

```
spacewalk-common-channels \
-a <architecture> \
<base_channel_name> \
<child_channel_name_1> \
<child_channel_name_2> \
... <child_channel_name_n>
```

2. **automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 계속하기 전에 동기화가 완료되었는지 확인합니다.



- spacewalk-common-channels** 가 제공하는 클라이언트 도구 채널은 SUSE가 아닌 Uyuni가 공급합니다.

모듈형 채널을 사용하는 경우 클라이언트에서 Python 3.6 모듈 스트림을 활성화해야 합니다. Python 3.6이 제공되지 않으면 **spacecmd** 패키지의 설치가 실패합니다.



- AppStream 채널에서 사용할 수 있는 패키지의 수가 업스트림과 Uyuni 채널 간에 서로 약간 불일치함을 알 수 있습니다. 또한 다른 시점에 추가한 동일 채널을 비교하면 그 수가 다른 것을 알 수 있습니다. 이는 CentOS가 리포지토리를 관리하는 방식에 원인이 있습니다. CentOS는 새 버전이 릴리스되면 이전 버전의 패키지를 제거하지만 Uyuni는 사용 기간과 관계없이 모두 그대로 유지합니다.



- AppStream 리포지토리는 모듈형 패키지를 제공합니다. 이로 인해 Uyuni Web UI에 부정확한 패키지 정보가 표시됩니다. Web UI 또는 API를 사용해 모듈형 리포지토리에서 직접 설치하거나 업그레이드하는 등의 패키지 작업을 수행할 수 없습니다.
- 또는 Salt 상태를 사용해 Salt 클라이언트에서 모듈형 패키지를 관리하거나 클라이언트에서 **dnf** 명령을 사용할 수 있습니다. CLM에 대한 자세한 내용은 **Administration > Content-lifecycle**을 참조하십시오.

4.7.1.2. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

1. Uyuni Web UI에서 **소프트웨어 > 관리 > 채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
2. **리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

4.7.1.3. 활성화 키 생성

CentOS 채널에 연결된 활성화 키를 생성해야 합니다.

활성화 키에 대한 자세한 내용은 **Client-configuration > Activation-keys**를 참조하십시오.

4.7.1.4. GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.



Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

GPG 키에 대한 자세한 내용은 [Client-configuration > Gpg-keys](#)에서 확인할 수 있습니다.

4.7.1.5. 클라이언트 등록

CentOS 클라이언트는 기타 모든 클라이언트와 동일한 방식으로 등록됩니다. 자세한 내용은 [Client-configuration > Registration-overview](#)를 참조하십시오.

4.7.1.6. 정오표 관리

CentOS 클라이언트를 업데이트할 때 패키지에는 업데이트에 대한 메타데이터가 포함되지 않습니다. 타사 정오표 서비스를 사용해 이 정보를 얻을 수 있습니다.



CEFS의 원저자는 도움이 됐으면 하는 마음으로 최선을 다해 패치나 정오표를 제공하지만 정확성이나 최신 상태를 보장하지는 않습니다. 따라서 패치 날짜가 부정확할 수 있습니다. 한번은 게시 날짜가 한 달 이상 늦은 날짜로 표시된 적도 있습니다. 이러한 경우에 대한 자세한 내용은 <https://github.com/stevemeier/cefs/issues/28#issuecomment-656579382> 및 <https://github.com/stevemeier/cefs/issues/28#issuecomment-656573607>을 참조하십시오.

패치 데이터 관련 문제나 지연으로 인해 Uyuni 서버로 신뢰할 수 없는 패치 정보가 임포트될 수 있습니다. 이로 인해 보고서, 감사, CVE 업데이트 또는 기타 패치 관련 정보가 부정확해질 수도 있습니다. 패치 데이터를 독립적으로 확인하거나 보안 관련 요구사항 및 인증 기준에 따라 다른 운영 체제를 선택하는 등 이 서비스 사용을 대체할 수 있는 수단을 고려해 보시기 바랍니다.

절차: 정오표 서비스 설치

- Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **sle-module-development-tools** 모듈을 추가합니다.

```
SUSEConnect --product sle-module-development-tools/15.2/x86_64
```

- 다음과 같이 정오표 서비스 종속성을 설치합니다.

```
zypper in perl-Text-Unidecode
```

- /etc/rhn/rhn.conf**에서 다음과 같은 줄을 추가하거나 편집합니다.

```
java.allow_adding_patches_via_api = centos7-updates-x86_64,centos7-
x86_64,centos7-extras-x86_64
```

4. Tomcat을 다시 시작합니다.

```
systemctl restart tomcat
```

5. 다음과 같이 정오표 스크립트에 대해 파일을 생성합니다.

```
touch /usr/local/bin/cent-errata.sh
```

6. 이 스크립트를 포함할 새 파일을 편집하고, 필요에 따라 리포지토리 상세 정보를 편집합니다. 이 스크립트는 외부 정오표 서비스에서 정오표 상세 정보를 가져와 압축을 풀고 상세 정보를 게시합니다.

```
#!/bin/bash
mkdir -p /usr/local/centos
cd /usr/local/centos
rm *.xml
wget -c http://cefs.steve-meier.de/errata.latest.xml
#wget -c https://www.redhat.com/security/data/oval/com.redhat.rhsa-
all.xml
wget -c https://www.redhat.com/security/data/oval/com.redhat.rhsa-
RHEL7.xml.bz2
bzip2 -d com.redhat.rhsa-RHEL7.xml.bz2
wget -c http://cefs.steve-meier.de/errata-import.tar
tar xvf errata-import.tar
chmod +x /usr/local/centos/errata-import.pl
export SPACEWALK_USER='<adminname>' ; export SPACEWALK_PASS='<password>'
/usr/local/centos/errata-import.pl --server '<servername>' \
--errata /usr/local/centos/errata.latest.xml \
--include-channels=centos7-updates-x86_64,centos7-x86_64,centos7-
-extras-x86_64 \
--publish --rhsa-oval /usr/local/centos/com.redhat.rhsa-RHEL7.xml
```

7. 다음과 같이 cron 작업을 설정하여 매일 스크립트를 실행합니다.

```
ln -s /usr/local/bin/cent-errata.sh /etc/cron.daily
```

이 도구에 대한 자세한 내용은 <https://cefs.steve-meier.de/>를 참조하십시오.

4.8. Debian 클라이언트 등록

Debian 클라이언트를 Uyuni 서버에 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 달라집니다.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.

4.8.1. Debian 클라이언트 등록

이 섹션에는 Debian 운영 체제를 실행하는 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

Salt 클라이언트에 대해서만 Debian을 지원합니다. 기존 클라이언트는 지원하지 않습니다.

Debian 클라이언트에서는 부트스트래핑을 사용해 초기 상태 실행을 수행하고 프로파일을 업데이트할 수 있습니다.



- SUSE는 Debian 운영 체제에 대한 지원을 제공하지 않습니다. Uyuni를(를) 통해 Debian 클라이언트를 관리할 수 있지만 지원은 제공되지 않습니다. Uyuni를(를) 사용한 Debian 클라이언트 관리는 현재 시험 중입니다. 이 지침은 Debian 10 및 Debian 11에서 테스트를 완료하였습니다. 프로덕션 환경에서는 Debian 클라이언트를 사용하지 마십시오.

4.8.1.1. 등록 준비

Debian 클라이언트를 Uyuni 서버에 등록하려면 다음과 같은 약간의 준비가 필요합니다.

- DNS가 올바르게 구성되어 있는지 확인하고 클라이언트에 항목을 제공합니다. 또는 Uyuni 서버와 클라이언트에서 적절한 항목으로 `/etc/hosts` 파일을 구성할 수 있습니다.
- 클라이언트의 날짜 및 시간을 Uyuni 서버와 동기화한 후 등록해야 합니다.

4.8.1.2. 소프트웨어 채널 추가

Debian 클라이언트를 Uyuni 서버에 등록하려면 먼저 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.



- 다음 섹션에서 설명은 종종 **x86_64** 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

이 절차에 필요한 채널은 다음과 같습니다.

표 29. Debian 채널 - CLI

OS 버전	기본 채널	클라이언트 채널	업데이트 채널	보안 채널
Debian 10	debian-10-pool-amd64-uyuni	debian-10-amd64-uyuni-client	debian-10-amd64-main-updates-uyuni	debian-10-amd64-main-security-uyuni

OS 버전	기본 채널	클라이언트 채널	업데이트 채널	보안 채널
Debian 11	debian-11-pool-amd64-uyuni	debian-11-amd64-uyuni-client	debian-11-amd64-main-updates-uyuni	debian-11-amd64-main-security-uyuni

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root로 **spacewalk-common-channels** 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- automatic synchronization이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

- 계속하기 전에 동기화가 완료되었는지 확인합니다.

4.8.1.3. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

- Uyuni Web UI에서 **소프트웨어** > **관리** > **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
- 리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

- Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

- 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.



Debian 채널은 규모가 매우 클 수 있습니다. 때로 동기화에 몇 시간이 걸릴 수 있습니다.

4.8.1.4. GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.



Trusting a GPG key is important for security on clients. It is the task of the

administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

GPG 키에 대한 자세한 내용은 [Client-configuration > Gpg-keys](#)에서 확인할 수 있습니다.



- Debian 클라이언트를 설치하려면 여러 개의 GPG 키가 필요할 수 있습니다.

타사 Debian 리포지토리를 동기화하는 경우 서버에서 적절한 GPG 키를 가져와야 합니다. GPG 키가 누락되면 동기화에 실패합니다.

Debian 리포지토리의 경우, 메타데이터만 서명됩니다. 그러므로 소프트웨어 채널용 GPG 키를 가져올 필요가 없습니다. Uyuni는 패키지에 다시 서명하지 않습니다.

Uyuni 서버로 이미 가져온 GPG 키를 살펴보려면 다음 명령을 실행하십시오.

```
sudo gpg --homedir /var/lib/spacewalk/gpgdir --list-keys
```

새 GPG 키를 가져오려면 **--import** 파라미터를 사용하십시오.

```
sudo gpg --homedir /var/lib/spacewalk/gpgdir --import <filename>.gpg
```

4.8.1.5. 루트 액세스

Debian의 루트 사용자는 SSH 액세스에 대해 기본적으로 비활성화되어 있습니다.

일반 사용자를 사용하여 온보딩하려면 **sudoers** 파일을 편집해야 합니다.

절차: 루트 사용자에게 액세스 권한 부여

- 클라이언트에서 다음과 같이 **sudoers** 파일을 편집합니다.

```
sudo visudo
```

이 줄을 **sudoers** 파일 끝에 추가하여 사용자에게 **sudo** 액세스 권한을 부여합니다. 다음과 같이 **<user>**를 Web UI에서 클라이언트를 부트스트래핑하고 있는 사용자의 이름으로 교체합니다.

```
<user>  ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2,  
/usr/bin/python3, /var/tmp/venv-salt-minion/bin/python
```



이 절차는 클라이언트 등록에 필요한 비밀번호가 없어도 루트에 액세스 권한을 부여합니다. 클라이언트는 성공적으로 설치된 후 루트 권한으로 실행되므로 액세스 권한이 더 이상 필요 없습니다. 클라이언트가 성공적으로 설치된 후 **sudoers** 파일에서 이 줄을 제거하는 것이 좋습니다.

4.8.1.6. 클라이언트 등록

Debian 클라이언트를 등록하려면 부트스트랩 리포지토리가 필요합니다. 기본적으로 부트스트랩 리포지토리는 매일 재생성됩니다. 명령 프롬프트에서 다음 명령을 사용해 부트스트랩 리포지토리를 수동으로 생성할 수 있습니다.

```
mgr-create-bootstrap-repo
```

Debian 10의 경우 프롬프트가 표시되면 **debian10-amd64-uyuni**를 선택합니다.

클라이언트 등록에 대한 자세한 내용은 [Client-configuration > Registration-overview](#)에서 참조하십시오.

4.9. Oracle 클라이언트 등록

Oracle Linux 클라이언트를 Uyuni 서버에 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 달라집니다.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.

4.9.1. Oracle Linux 클라이언트 등록

이 섹션에는 Oracle Linux 운영 체제를 실행하는 기존 및 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

Traditional clients are not available on Oracle Linux 9 and 8. Oracle Linux 9 and Oracle Linux 8 clients are only supported as Salt clients.

4.9.1.1. 소프트웨어 채널 추가

Oracle Linux 클라이언트를 Uyuni 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

현재 지원되는 아키텍처는 **x86_64** 및 **aarch64**입니다. 지원되는 제품 및 아키텍처의 전체 목록은 [Client-configuration > Supported-features](#)에서 참조하십시오.



- 다음 섹션에서 설명은 종종 **x86_64** 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

이 절차에 필요한 채널은 다음과 같습니다.

표 30. Oracle 채널 - CLI

OS 버전	기본 채널	클라이언트 채널	업데이트 채널
Oracle Linux 9	oraclelinux9	oraclelinux9-uyuni-client	oraclelinux9-appstream

OS 버전	기본 채널	클라이언트 채널	업데이트 채널
Oracle Linux 8	oraclelinux8	oraclelinux8-uyuni-client	oraclelinux8-appstream
Oracle Linux 7	oraclelinux7	oraclelinux7-uyuni-client	-

명령 프롬프트에서 소프트웨어 채널 추가

- Uyuni 서버의 명령 프롬프트에서 root로 **spacewalk-common-channels** 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

- automatic synchronization**이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

- 계속하기 전에 동기화가 완료되었는지 확인합니다.



spacewalk-common-channels 가 제공하는 클라이언트 도구 채널은 SUSE가 아닌 Uyuni가 공급합니다.



Oracle Linux 9 및 Oracle Linux 8 클라이언트의 경우 기본 채널과 AppStream 채널을 모두 추가하십시오. 사용자에게는 두 채널의 패키지가 모두 필요합니다. 두 채널을 모두 추가하지 않으면 패키지가 누락되어 부트스트랩 리포지토리를 생성할 수 없습니다.

모듈형 채널을 사용하는 경우 클라이언트에서 Python 3.6 모듈 스트림을 활성화해야 합니다. Python 3.6이 제공되지 않으면 **spacecmd** 패키지의 설치가 실패합니다.



AppStream 리포지토리는 모듈형 패키지를 제공합니다. 이로 인해 Uyuni Web UI에 부정확한 패키지 정보가 표시됩니다. Web UI 또는 API를 사용해 모듈형 리포지토리에서 직접 설치하거나 업그레이드하는 등의 패키지 작업을 수행할 수 없습니다.

또는 Salt 상태를 사용해 Salt 클라이언트에서 모듈형 패키지를 관리하거나 클라이언트에서 **dnf** 명령을 사용할 수 있습니다. CLM에 대한 자세한 내용은 **Administration > Content-lifecycle**을 참조하십시오.

4.9.1.2. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

- Uyuni Web UI에서 **소프트웨어**, **관리**, **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.

2. 리포지토리 탭으로 이동한 다음, 동기화를 클릭하고 동기화 상태를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 tail 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

4.9.1.3. 활성화 키 생성

Oracle Linux 채널에 연결된 활성화 키를 생성해야 합니다.

활성화 키에 대한 자세한 내용은 **Client-configuration > Activation-keys**를 참조하십시오.

4.9.1.4. GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.



Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

GPG 키에 대한 자세한 내용은 **Client-configuration > Gpg-keys**에서 확인할 수 있습니다.



Oracle Linux 9 및 Oracle Linux 8 클라이언트의 경우

```
o18-gpg-pubkey-82562EA9AD986DA3.key
```

Oracle Linux 7 클라이언트의 경우

```
o167-gpg-pubkey-72F97B74EC551F0A3.key
```

4.9.1.5. 클라이언트 등록

Oracle Linux 클라이언트는 기타 모든 클라이언트와 동일한 방식으로 등록됩니다. 자세한 내용은 **Client-configuration > Registration-overview**를 참조하십시오.

4.10. Red Hat 클라이언트 등록

Red Hat 컨텐트 제공 네트워크(CDN) 또는 Red Hat 업데이트 인프라(RHUI)를 사용해 Uyuni 서버에 Red Hat Enterprise Linux 클라이언트를 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 달라집니다.

시작하기 전에 클라이언트의 날짜 및 시간이 Uyuni 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [Client-configuration > Activation-keys](#)를 참조하십시오.

4.10.1. Red Hat Enterprise Linux 클라이언트를 CDN으로 등록

This section contains information about using the Red Hat content delivery network (CDN) to register traditional and Salt clients running Red Hat Enterprise Linux operating systems.

Traditional clients are available on Red Hat Enterprise Linux 7 only. Red Hat Enterprise Linux 8 and Red Hat Enterprise Linux 9 clients are supported as Salt clients.

대신에 Red Hat 업데이트 인프라(RHUI) 사용법에 대한 정보는 [Client-configuration > Clients-rh-rhui](#)를 참조하십시오.



You are responsible for arranging access to Red Hat base media repositories and RHEL installation media, as well as connecting Uyuni Server to the Red Hat content delivery network. You must obtain support from Red Hat for all your RHEL systems. If you do not do this, you might be violating your terms with Red Hat.

4.10.1.1. 자격 및 인증서 임포트

Red Hat 클라이언트는 Red Hat 인증 주체(CA)와 자격 인증서 및 자격 키가 있어야 합니다.

자격 인증서에는 지원 구독 기간과 일치하는 만료일이 포함되어 있습니다. 중단을 방지하려면 지원 구독 기간이 종료되는 시점이 될 때마다 이 프로세스를 반복해야 합니다.

Red Hat은 구독 할당을 관리할 수 있도록 구독 관리자 도구를 제공합니다. 이 도구는 로컬로 실행되어 설치된 제품 및 구독을 추적합니다. 클라이언트는 구독 관리자로 등록되어 있어야 인증서를 취득할 수 있습니다.

Red Hat 클라이언트는 URL을 사용해 리포지토리를 복제합니다. URL은 Red Hat 클라이언트가 어디에 등록되었는지에 따라 달라집니다.

Red Hat 클라이언트는 다음 세 가지 방식으로 등록할 수 있습니다.

- redhat.com의 Red Hat 컨텐트 제공 네트워크(CDN)
- Red Hat Satellite 서버
- 클라우드의 Red Hat 업데이트 인프라(RHUI)

이 안내서에서는 Red Hat CDN에 등록된 클라이언트에 대해 설명합니다. CDN에 등록된 시스템이 최소 한 개 있어야 하고, 이와 함께 리포지토리 컨텐트에 대한 인증 구독이 있어야 합니다.

대신에 Red Hat 업데이트 인프라(RHUI) 사용법에 대한 정보는 [Client-configuration > Clients-rh-rhui](#)를 참조하십시오.



클라이언트 시스템에 대한 Satellite 서버는 Satellite 서버 및 구독이 있어야 합니다. Satellite 인증서를 사용하는 클라이언트는 Uyuni 서버에서 지원하지 않습니다.



- 자격 인증서에는 지원 구독 기간과 일치하는 만료일이 포함되어 있습니다. 중단을 방지하려면 지원 구독 기간이 종료되는 시점이 될 때마다 이 프로세스를 반복해야 합니다.

Red Hat은 구독 할당을 관리할 수 있도록 구독 관리자 도구를 제공합니다. 이 도구는 클라이언트 시스템에서 로컬로 실행되어 설치된 제품 및 구독을 추적합니다. 구독 관리자로 redhat.com에 등록한 후, 이 절차에 따라 인증서를 취득하십시오.

절차: 클라이언트를 구독 관리자에 등록

- 클라이언트 시스템의 명령 프롬프트에서 다음과 같이 구독 관리자 도구로 등록합니다.

```
subscription-manager register
```

프롬프트가 표시되면 Red Hat 포털 사용자 이름 및 비밀번호를 입력합니다.

- 명령 실행:

```
subscription-manager activate
```

- 다음과 같이 클라이언트 시스템에서 Uyuni 서버가 액세스할 수 있는 위치로 자격 인증서 및 키를 복사합니다.

```
cp /etc/pki/entitlement/ <example>/entitlement
```



- 자격 인증서 및 키에는 **.pem**이라는 파일 확장자가 있습니다. 또한 키는 파일 이름에 **key**가 포함되어 있습니다.

- 다음과 같이 클라이언트 시스템에서 자격 인증서 및 키와 동일한 웹 위치로 Red Hat CA 인증서 파일을 복사합니다.

```
cp /etc/rhsm/ca/redhat-uep.pem <example>/entitlement
```

Red Hat 클라이언트에서 리포지토리를 관리하려면 CA 및 자격 인증서를 Uyuni 서버로 임포트해야 합니다. 이렇게 하려면 임포트 절차를 세 차례 수행하여 자격 인증서, 자격 키, Red Hat 인증서 각각에 대해 하나씩 세 항목을 생성해야 합니다.

절차: 인증서를 서버에 임포트

- Uyuni 서버 Web UI에서 **시스템**, **자동 설치**, **GPG 및 SSL 키**로 이동합니다.
- 저장된 키/인증서 생성**을 클릭하고 자격 인증서에 대해 다음 파라미터를 설정합니다.
 - 설명** 필드에 **Entitlement-Cert-date**를 입력합니다.
 - 유형** 필드에서 **SSL**을 선택합니다.
 - 업로드할 파일을 선택해주십시오** 필드에서 자격 인증서를 저장한 위치로 이동하여 **.pem** 인증서 파일을 선택합니다.
- 키 생성**을 클릭합니다.

4. 저장된 키/인증서 생성을 클릭하고 자격 키에 대해 다음 파라미터를 설정합니다.
 - 설명 필드에 Entitlement-key-date를 입력합니다.
 - 유형 필드에서 SSL을 선택합니다.
 - 업로드할 파일을 선택해주십시오 필드에서 자격 키를 저장한 위치로 이동하여 .pem 키 파일을 선택합니다.
5. 키 생성을 클릭합니다.
6. 저장된 키/인증서 생성을 클릭하고 Red Hat 인증서에 대해 다음 파라미터를 설정합니다.
 - 설명 필드에 redhat-uep를 입력합니다.
 - 유형 필드에서 SSL을 선택합니다.
 - 업로드할 파일을 선택해주십시오 필드에서 Red Hat 인증서를 저장한 위치로 이동하여 인증서 파일을 선택합니다.
7. 키 생성을 클릭합니다.

4.10.1.2. 소프트웨어 채널 추가

Red Hat 클라이언트를 Uyuni 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.



- 다음 섹션에서 설명은 종종 x86_64 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

이 절차에 필요한 채널은 다음과 같습니다.

표 31. Red Hat 채널- CLI

OS 버전	기본 채널	클라이언트 채널	도구 채널
Red Hat 7	rhel-x86_64-server-7	-	res7-suse-manager-tools-x86_64
Red Hat 8	rhel8-pool-x86_64	-	res8-manager-tools-pool-x86_64
Red Hat 9	el9-pool-x86_64	-	el9-manager-tools-pool-x86_64, el9-manager-tools-updates-x86_64

명령 프롬프트에서 소프트웨어 채널 추가

1. Uyuni 서버의 명령 프롬프트에서 root로 spacewalk-common-channels 명령을 사용하여 적절한 채널을 추가합니다.

```
spacewalk-common-channels \
<base_channel_label> \
<child_channel_label_1> \
<child_channel_label_2> \
... <child_channel_label_n>
```

2. [automatic synchronization](#)이 꺼진 경우 다음 채널을 동기화하십시오.

```
spacewalk-repo-sync -p <base_channel_label>
```

3. 계속하기 전에 동기화가 완료되었는지 확인합니다.



spacewalk-common-channels 가 제공하는 클라이언트 도구 채널은 SUSE가 아닌 Uyuni가 공급합니다.



AppStream 리포지토리는 모듈형 패키지를 제공합니다. 이로 인해 Uyuni Web UI에 부정확한 패키지 정보가 표시됩니다. Web UI 또는 API를 사용해 모듈형 리포지토리에서 직접 설치하거나 업그레이드하는 등의 패키지 작업을 수행할 수 없습니다.

또는 Salt 상태를 사용해 Salt 클라이언트에서 모듈형 패키지를 관리하거나 클라이언트에서 `dnf` 명령을 사용할 수 있습니다. CLM에 대한 자세한 내용은 [Administration > Content-lifecycle](#)을 참조하십시오.

4.10.1.3. 사용자 정의 리포지토리 및 채널 준비

Red Hat CDN에서 소프트웨어를 미러링하려면 URL로 CDN에 링크된 사용자 정의 채널 및 리포지토리를 Uyuni에 생성해야 합니다. 이 작업이 제대로 수행되려면 Red Hat 포털에 이러한 제품에 대한 자격이 있어야 합니다. 미러링하려는 리포지토리의 URL을 다음과 같이 구독 관리자 도구를 사용해 가져올 수 있습니다.

```
subscription-manager repos
```

이 리포지토리 URL을 사용해 사용자 정의 리포지토리를 생성할 수 있습니다. 이렇게 하면 클라이언트를 관리하는 데 필요한 컨텐트만 미러링할 수 있습니다.



Red Hat 포털에 올바른 자격이 있는 경우 Red Hat 리포지토리의 사용자 정의 버전만 생성할 수 있습니다.

이 절차에 필요한 상세 정보는 다음과 같습니다.

표 32. Red Hat 사용자 정의 리포지토리 설정

옵션	설정
리포지토리 URL	Red Hat CDN이 제공하는 컨텐트 URL
서명 메타데이터가 있습니까?	모든 Red Hat Enterprise 리포지토리를 선택 취소
SSL CA 인증서	<code>redhat-uep</code>
SSL 클라이언트 인증서	<code>Entitlement-Cert-date</code>
SSL 클라이언트 키	<code>Entitlement-Key-date</code>

절차: 사용자 정의 리포지토리 생성

1. Uyuni 서버 Web UI에서 **소프트웨어 > 관리 > 리포지토리**로 이동합니다.

2. **리포지토리 생성**을 클릭하여 리포지토리에 적절한 파라미터를 설정합니다.
3. **리포지토리 생성**을 클릭합니다.
4. 생성해야 하는 모든 리포지토리에 대해 이를 반복합니다.

이 절차에 필요한 채널은 다음과 같습니다.

표 33. Red Hat 사용자 정의 채널

OS Version	Base Product	Base Channel
Red Hat 7	RHEL7 Base x86_64	rhel7-pool-x86_64

절차: 사용자 정의 채널 생성

1. Uyuni 서버 Web UI에서 **소프트웨어** > **관리** > **채널**로 이동합니다.
2. **채널 생성**을 클릭하여 채널에 적절한 파라미터를 설정합니다.
3. **상위 채널** 필드에서 적절한 기본 채널을 선택합니다.
4. **채널 생성**을 클릭합니다.
5. 생성해야 하는 모든 채널에 대해 이를 반복합니다. 각 사용자 정의 리포지토리에는 하나의 사용자 정의 채널이 있어야 합니다.

소프트웨어 > **채널 목록** > **전체**로 이동하여 적절한 채널 및 리포지토리를 모두 생성했는지 확인할 수 있습니다.



For Red Hat 8 clients, add both the Base and AppStream channels. You require packages from both channels. If you do not add both channels, you cannot create the bootstrap repository, due to missing packages.

If you are using modular channels, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the **spacecmd** package will fail.
include::snippets/manual_associate.adoc[]

4.10.1.4. 동기화 상태 확인

절차: Web UI에서 동기화 진행률 확인

1. Uyuni Web UI에서 **소프트웨어** > **관리** > **채널**로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
2. **리포지토리** 탭으로 이동한 다음, **동기화**를 클릭하고 **동기화 상태**를 확인합니다.

절차: 명령 프롬프트에서 동기화 진행 상태 확인

1. Uyuni 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 **tail** 명령을 사용해 동기화 로그 파일을 확인합니다.

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

2. 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다. 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.



Red Hat Enterprise Linux channels can be very large. Synchronization can

有时候会花费数小时。

절차: 옵션: Salt 상태를 생성하여 구성 파일 배포

1. Uyuni 서버 Web UI에서 **구성 > 채널**로 이동합니다.
2. **상태 채널 생성**을 클릭합니다.
 - 이름 필드에 **subscription-manager: disable yum plugins**를 입력합니다.
 - 레이블 필드에 **subscription-manager-disable-yum-plugins**를 입력합니다.
 - 설명 필드에 **subscription-manager: disable yum plugins**를 입력합니다.
 - SLS 컨텐트 필드를 빈 상태로 둡니다.
3. **구성 채널 생성**을 클릭합니다.
4. **구성 파일 생성**을 클릭합니다.
 - 파일 이름/경로 필드에 **/etc/yum/pluginconf.d/subscription-manager.conf**를 입력합니다.
 - 파일 내용 필드에 다음과 같이 입력합니다.
 - ---- enabled=0

```
. btn:[구성 파일 생성]을 클릭합니다.
. [guimenu]``Salt 파일 시스템 경로`` 필드의 값을 적어 둡니다.
. 구성 채널의 이름을 클릭합니다.
. [guimenu]``'init.sls' 파일 보기/편집``을 클릭합니다.
* [guimenu]``파일 내용`` 필드에 다음과 같이 입력합니다.
+
* ----
configure_subscription-manager-disable-yum-plugins:
cmd.run:
  - name: subscription-manager config --rhsm.auto_enable_yum_plugins=0
  - watch:
    - file: /etc/yum/pluginconf.d/subscription-manager.conf
file.managed:
  - name: /etc/yum/pluginconf.d/subscription-manager.conf
  - source: salt://etc/yum/pluginconf.d/subscription-manager.conf
  ----
. btn:[구성 파일 업데이트]를 클릭합니다.
```

[NOTE]

=====

The ``Creating a Salt State to Deploy Configuration Files`` procedure is optional.

=====

. 절차: {rhel} 클라이언트에 대해 시스템 그룹 생성

- . {productname} 서버 {webui}에서 menu:시스템[시스템 그룹]으로 이동합니다.
- . btn:[그룹 생성]을 클릭합니다.
- * [guimenu]``이름`` 필드에 [systemitem]``rhel-systems``라고 입력합니다.
- * [guimenu]``설명`` 필드에 [systemitem]``모든 RHEL 시스템``이라고 입력합니다.
- . btn:[그룹 생성]을 클릭합니다.
- . [guimenu]``상태`` 탭을 클릭합니다.
- . [guimenu]``구성 채널`` 탭을 클릭합니다.
- . 검색 상자에 [systemitem]``subscription-manager: disable yum plugins``라고 입력합니다.
- . btn:[검색]을 클릭하여 상태를 확인합니다.
- . [systemitem]``할당`` 열에서 상태의 확인란을 클릭합니다.
- . btn:[변경 사항 저장]을 클릭합니다.

Click `btn:[Confirm]`. If you already have RHEL systems added to {productname}, assign them to the new system group, and then apply the `highstate`.

- . 절차: 시스템 그룹을 활성화 키에 추가

위에서 생성한 시스템 그룹을 포함하려면 RHEL 시스템에 사용한 활성화 키를 수정해야 합니다.

- . {productname} 서버 {webui}에서 menu:시스템[활성화 키]로 이동합니다.
- . RHEL 시스템에 사용한 각 활성화 키를 클릭하고 다음과 같이 합니다.
- . [guimenu]``그룹`` 탭의 [guimenu]``조인`` 하위 탭으로 이동합니다.
- . [systemitem]``rhel-systems 선택``의 확인란을 선택합니다.
- . btn:[선택한 그룹 조인]을 클릭합니다.

== GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.

[IMPORTANT]

=====

Trusting a GPG key is important for security on clients.
It is the task of the administrator to decide which keys are needed and can be trusted.

Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

=====

GPG 키에 대한 자세한 내용은 `xref:client-configuration:gpg-keys.adoc[]`에서 확인할 수 있습니다.

== 클라이언트 등록

To register your {redhat} clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

`mgr-create-bootstrap-repo`

클라이언트 등록에 대한 자세한 내용은 [xref:client-configuration:registration-overview.adoc](#)[]에서 참조하십시오.

```
//[WARNING]
//=====
//To register and use {rhel}{nbsp}6 clients, you need to configure the
{productname} Server to support older types of SSL encryption.
//For more information about how to resolve this error, see ``Registering
Older Clients`` at xref:administration:troubleshooting/tshoot-intro.adoc[Troubleshooting].
//=====

:leveloffset!:
:leveloffset: +3

[[clients-rh-rhui]]
= {rhel} 클라이언트를 RHUI로 등록

// SUSE Liberty Linux not available at Uyuni for now
```

This section contains information about using {redhat} update infrastructure (RHUI) to register traditional and Salt clients running {rhel} operating systems.

Traditional clients are available on {rhel}{nbsp}7 only. {rhel}{nbsp}8 and {rhel}{nbsp}9 clients are supported as Salt clients.

Amazon EC2와 같은 공용 클라우드에서 클라이언트를 실행하는 경우 이 방법을 사용하십시오.

RHUI를 {redhat} 컨텐트 제공 네트워크(CDN)와 함께 사용하여 {rhel} 구독을 관리할 수 있습니다. {redhat} CDN 사용에 대한 정보는 [xref:client-configuration:clients-rh-cdn.adoc](#)[]을 참조하십시오.

[IMPORTANT]

=====

```
// SUSE Liberty Linux not available at Uyuni for now
```

You are responsible for connecting {productname} Server to the {redhat} update infrastructure. All clients that get updates using this RHUI certificate need to be correctly licensed, please check with your cloud provider and the {redhat} terms of service for more information.

=====

[NOTE]

=====

RHUI로 등록한 {rhel} 클라이언트가 켜져 있지 않은 경우 {redhat}은 인증서가 잘못되었다고 선언할 수 있습니다. 이 경우 클라이언트를 다시 켜거나 새 RHUI 인증서를 가져와야 합니다.

=====

== 자격 및 인증서 임포트

{redhat} 클라이언트는 {redhat} 인증 주체(CA)와 자격 인증서 및 자격 키가 있어야 합니다.

{redhat} 클라이언트는 URL을 사용해 리포지토리를 복제합니다. URL은 {redhat} 클라이언트가 어디에 등록되었는지에 따라 달라집니다.

{redhat} 클라이언트는 다음 세 가지 방식으로 등록할 수 있습니다.

- * redhat.com의 {redhat} 컨텐트 제공 네트워크(CDN)
- * {redhat} Satellite 서버
- * 클라우드의 {redhat} 업데이트 인프라(RHUI)

이 안내서에서는 {redhat} 업데이트 인프라(RHUI)에 등록된 클라이언트에 대해 설명합니다. RHUI에 등록된 시스템이 최소 한 개 있어야 하고, 이와 함께 리포지토리 컨텐트에 대한 인증 구독이 있어야 합니다.

{redhat} 컨텐트 제공 네트워크(CDN)를 사용하는 방법에 대한 정보 대신에 [xref:client-configuration:clients-rh-cdn.adoc](#)[]을 참조하십시오.

[IMPORTANT]

=====

클라이언트 시스템에 대한 Satellite 서버는 Satellite 서버 및 구독이 있어야 합니다. Satellite 인증서를 사용하는 클라이언트는 {productname} 서버에서 지원하지 않습니다.

=====

The entitlement certificates and keys need to be copied from the client system to a location that your web browser can access.

키 및 인증서의 이름은 여기에 표시되는 이름과 약간 다를 수 있습니다. 자격 인증서 및 {redhat} CA 인증서 파일의 파일 확장자는 [path]``.crt``입니다. 키의 파일 확장자는 [path]``.key``입니다.

. 절차: 인증서를 서버에 복사

. Copy your entitlement certificate and key from the client system, to your workstation:

+

Amazon EC2::

+

```
cp /etc/pki/rhui/product/content-<version>.crt /<example>/entitlement/ cp /etc/pki/rhui/content-<version>.key /<example>/entitlement/
```

+

Azure::

+

. 다음 명령을 사용하여 인증서 체인을 확인합니다.

+

```
openssl s_client -connect rhui-1.microsoft.com:443 -showcerts
```

+

샘플 출력은 다음과 같습니다.

+

```
CONNECTED(00000003) depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert Global Root G2 verify return:1 depth=1 C = US, O = Microsoft Corporation, CN = Microsoft Azure TLS Issuing CA 06 verify return:1 depth=0 C = US, ST = WA, L = Redmond, O = Microsoft Corporation, CN = rhui-1.microsoft.com verify return+
```

+

Check the second certificate ([literal]``CN = Microsoft Azure``), if it is the same on your VM, note the certificate name. Refer to the <https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/certificateAuthorities> to download the certificate.

+

- . Click the AIA link to download the certificate. The certificate will be downloaded with the [literal]``.cer`` suffix.
- + . Convert it to [literal]``.crt`` with the command:
- +

```
openssl x509 -inform DER -in <example.cer> -out <example.crt>
```

- + Google Cloud Platform::
- +

```
cp /etc/pki/rhui/product/content.crt /<example>/entitlement/ cp /etc/pki/rhui/key.pem /<example>/entitlement/
```

- + . 다음과 같이 클라이언트 시스템에서 자격 인증서 및 키와 동일한 위치로 {redhat} CA 인증서 파일을 복사합니다.
- + Amazon EC2::
- +

```
cp /etc/pki/rhui/cdn.redhat.com-chain.crt /<example>/entitlement
```

- + Azure::
- + * 변환된 인증서를 /<example>/entitlement에 업로드합니다.
- + Google Cloud Platform::
- +

```
cp /etc/pki/rhui/ca.crt /<example>/entitlement
```

To manage repositories on your {redhat} client, you need to import the CA and entitlement certificates to the {productname} Server.

This requires that you perform the import procedure three times, to create three entries, one of each for the entitlement certificate, the entitlement key, and the {redhat} certificate.

. 절차: 인증서를 서버에 임포트

- . {productname} 서버 {webui}에서 menu:시스템[자동 설치 > GPG 및 SSL 키]로 이동합니다.
- . btn:[저장된 키/인증서 생성]을 클릭하고 자격 인증서에 대해 다음 파라미터를 설정합니다.
 - * [guimenu]``설명`` 필드에 [systemitem]``Entitlement-Cert-Date``를 입력합니다.
 - * [guimenu]``유형`` 필드에서 [systemitem]``SSL``을 선택합니다.
 - * [guimenu]``업로드할 파일을 선택해주십시오`` 필드에서 자격 인증서를 저장한 위치로 이동하여 [path]``.crt`` 인증서 파일을 선택합니다.
- . btn:[키 생성]을 클릭합니다.
- . btn:[저장된 키/인증서 생성]을 클릭하고 자격 키에 대해 다음 파라미터를 설정합니다.
 - * [guimenu]``설명`` 필드에 [systemitem]``Entitlement-Key-Date``를 입력합니다.
 - * [guimenu]``유형`` 필드에서 [systemitem]``SSL``을 선택합니다.
 - * [guimenu]``업로드할 파일을 선택해주십시오`` 필드에서 자격 키를 저장한 위치로 이동하여 [path]``.key`` 키 파일을 선택합니다.
- . btn:[키 생성]을 클릭합니다.
- . btn:[저장된 키/인증서 생성]을 클릭하고 {redhat} 인증서에 대해 다음 파라미터를 설정합니다.
 - * [guimenu]``설명`` 필드에 [systemitem]``redhat-cert``를 입력합니다.
 - * [guimenu]``유형`` 필드에서 [systemitem]``SSL``을 선택합니다.
 - * [guimenu]``업로드할 파일을 선택해주십시오`` 필드에서 {redhat} 인증서를 저장한 위치로 이동하여 인증서 파일을 선택합니다.
- . btn:[키 생성]을 클릭합니다.

== 소프트웨어 채널 추가

```
// 2022-04-21, ke:
// Section sequence according to https://github.com/uyuni-project/uyuni-
docs/pull/1535
```

{redhat} 클라이언트를 {productname} 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

[NOTE]

=====

다음 섹션에서 설명은 종종 [literal]``x86_64`` 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

=====

이 절차에 필요한 채널은 다음과 같습니다.

```
[[redhat-rhui-channels-cli]]
[cols="1,1,1,1", options="header"]
.Red Hat 채널- CLI
| ===

| OS Version
| Base Channel
| Client Channel
| Tools Channel

| {redhat} 9
| el9-pool-x86_64
| -
| el9-manager-tools-pool-x86_64, el9-manager-tools-updates-x86_64

| {redhat} 8
| rhel8-pool-x86_64
| -
| res8-manager-tools-pool-x86_64

| {redhat} 7
| rhel-x86_64-server-7
| -
| res7-suse-manager-tools-x86_64

| ===
```

.명령 프롬프트에서 소프트웨어 채널 추가

. {productname} 서버의 명령 프롬프트에서 root로 [command]``spacewalk-common-channels`` 명령을 사용하여 적절한 채널을 추가합니다.

+

```
spacewalk-common-channels \ <base_channel_label> \ <child_channel_label_1> \
<child_channel_label_2> \ ... <child_channel_label_n>
```

```
. xref:administration:custom-
channels.adoc#_custom_channel_synchronization[automatic synchronization]이
꺼진 경우 다음 채널을 동기화하십시오.
```

+

```
spacewalk-repo-sync -p <base_channel_label>
```

- . 계속하기 전에 동기화가 완료되었는지 확인합니다.

[NOTE]

=====

[command] ``spacewalk-common-channels`` 가 제공하는 클라이언트 도구 채널은 {suse}가 아닌 {uyuni}가 공급합니다.

=====

[IMPORTANT]

=====

AppStream 리포지토리는 모듈형 패키지를 제공합니다. 이로 인해 {productname} {webui}에 부정확한 패키지 정보가 표시됩니다. {webui} 또는 API를 사용해 모듈형 리포지토리에서 직접 설치하거나 업그레이드하는 등의 패키지 작업을 수행할 수 없습니다.

또는 Salt 상태를 사용해 Salt 클라이언트에서 모듈형 패키지를 관리하거나 클라이언트에서 [command] ``dnf`` 명령을 사용할 수 있습니다. CLM에 대한 자세한 내용은 [xref:administration:content-lifecycle.adoc](#) []을 참조하십시오.

=====

RHUI를 사용하려면 필요한 HTTP 헤더를 구성 파일에 수동으로 추가해야 합니다. 이 헤더가 없으면 클라이언트 동기화를 성공적으로 수행할 수 없습니다.

. 절차: HTTP 헤더를 구성 파일에 추가

. RHUI 인스턴스에서 [systemitem] ``X-RHUI-ID`` 및 [systemitem] ``X-RHUI-SIGNATURE`` HTTP 헤더를 찾습니다.

{redhat} 클라이언트에서 다음 명령을 사용해 [systemitem] ``169.254.169.254`` 의 클라우드 인스턴스 메타데이터 API에서 값을 가져올 수 있습니다.

+

```
echo "X-RHUI-ID=$(curl -s http://169.254.169.254/latest/dynamic/instance-identity/document|base64|tr -d '\n')"  
echo "X-RHUI-SIGNATURE=$(curl -s http://169.254.169.254/latest/dynamic/instance-identity/signature|base64|tr -d '\n')"
```

. [path] ``/etc/rhn/spacewalk-repo-sync/extrah_headers.conf`` 구성 파일을 열고, 다음과 같은 줄을 추가하거나 정확한 정보로 편집합니다.

+

[<channel_label_1>] X-RHUI-ID=<value> X-RHUI-SIGNATURE=<value>

[<channel_label_2>] X-RHUI-ID=<value> X-RHUI-SIGNATURE=<value>

+

. Replace [literal]``<channel_label_X>`` above with the names of the custom channels you are planning to create (see next section):

+

X-RHUI-ID=… X-RHUI-SIGNATURE=…

+

[NOTE]

=====

{rhel} may renew these headers at any point in time. In such a case, repeat the procedure to get the new HTTP headers in place.

=====

== 사용자 정의 리포지토리 및 채널 준비

RHUI에서 소프트웨어를 미러링하려면 URL로 RHUI에 링크된 사용자 정의 채널 및 리포지토리를 {productname}에 생성해야 합니다. 이 작업이 제대로 수행되려면 Red Hat 포털에 이러한 제품에 대한 자격이 있어야 합니다. 미러링하려는 리포지토리의 URL을 다음과 같이 yum 유ти리티를 사용해 가져올 수 있습니다.

yum repolist -v | grep baseurl

이 리포지토리 URL을 사용해 사용자 정의 리포지토리를 생성할 수 있습니다. 이렇게 하면 클라이언트를 관리하는 데 필요한 컨텐트만 미러링할 수 있습니다.

[IMPORTANT]

=====

{redhat} 포털에 올바른 자격이 있는 경우 {redhat} 리포지토리의 사용자 정의 버전만 생성할 수 있습니다.

=====

이 절차에 필요한 상세 정보는 다음과 같습니다.

[[redhat-rhui-repos-manual]]
[cols="1,1", options="header"]

.Red Hat 사용자 정의 리포지토리 설정

| ===

옵션	설정
리포지토리 URL	RHUI가 제공하는 컨텐트 URL
서명 메타데이터가 있습니까?	모든 {redhat} Enterprise 리포지토리를 선택 취소
SSL CA 인증서	[systemitem]``redhat-cert``
SSL 클라이언트 인증서	[systemitem]``Entitlement-Cert-Date``
SSL 클라이언트 키	[systemitem]``Entitlement-Key-Date``

| ===

.절차: 사용자 정의 리포지토리 생성

- . {productname} 서버 {webui}에서 menu:소프트웨어[관리 > 리포지토리]로 이동합니다.
- . btn:[리포지토리 생성]을 클릭하여 리포지토리에 적절한 파라미터를 설정합니다.
- . btn:[리포지토리 생성]을 클릭합니다.
- . 생성해야 하는 모든 리포지토리에 대해 이를 반복합니다.

이 절차에 필요한 채널은 다음과 같습니다.

```
[[redhat-rhui-channels-custom]]
[cols="1,1,1", options="header"]
.Red Hat 사용자 정의 채널
| ===

// SUSE Liberty Linux not available at Uyuni for now

// SUSE Liberty Linux not available at Uyuni for now
| {redhat} 9 | RHEL | el9-pool-x86_64 | {redhat} 8 | RHEL or CentOS 8
Base | rhel8-pool-x86_64

// SUSE Liberty Linux not available at Uyuni for now
| {redhat} 7 | RHEL7 Base x86_64 | rhel7-pool-x86_64

| ===
```

.절차: 사용자 정의 채널 생성

- . {productname} 서버 {webui}에서 menu:소프트웨어[관리 > 채널]로 이동합니다.
- . btn:[채널 생성]을 클릭하여 채널에 적절한 파라미터를 설정합니다.
- . [guimenu]``상위 채널`` 필드에서 적절한 기본 채널을 선택합니다.
- . btn:[채널 생성]을 클릭합니다.
- . 생성해야 하는 모든 채널에 대해 이를 반복합니다. 각 사용자 정의 리포지토리에는 하나의 사용자 정의 채널이 있어야 합니다.

`menu:소프트웨어[채널 목록 > 전체]`로 이동하여 적절한 채널 및 리포지토리를 모두 생성했는지 확인할 수 있습니다.

[IMPORTANT]

====

{redhat} 8 클라이언트의 경우 기본 채널과 AppStream 채널을 모두 추가하십시오.

사용자에게는 두 채널의 패키지가 모두 필요합니다. 두 채널을 모두 추가하지 않으면 패키지가 누락되어 부트스트랩 리포지토리를 생성할 수 없습니다.

====

모든 채널을 생성했으면 생성한 리포지토리를 다음과 같이 채널과 연결할 수 있습니다.

. 절차: 채널을 리포지토리와 연결

- . {productname} 서버 {webui}에서 `menu:소프트웨어[관리 > 채널]`로 이동한 다음, 연결할 채널을 클릭합니다.
- . [guimenu]``리포지토리`` 탭으로 이동하여 이 채널에 연결할 리포지토리의 확인란을 선택합니다.
- . `btn:[리포지토리 업데이트]`를 클릭하여 채널과 리포지토리를 연결합니다.
- . 연결하려는 모든 채널과 리포지토리에 대해 이를 반복합니다.
- . 옵션: [guimenu]``동기화`` 탭으로 이동하여 이 리포지토리의 동기화에 대해 정기적 일정을 설정합니다.
- . `btn:[지금 동기화]`를 클릭하여 즉시 동기화를 시작합니다.

== 동기화 상태 확인

. 절차: {webui}에서 동기화 진행률 확인

- . {productname} {webui}에서 `menu:소프트웨어[관리 > 채널]`로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
- . [guimenu]``리포지토리`` 탭으로 이동한 다음, [guimenu]``동기화``를 클릭하고 [systemitem]``동기화 상태``를 확인합니다.

. 절차: 명령 프롬프트에서 동기화 진행 상태 확인

- . {productname} 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 [command]``tail`` 명령을 사용해 동기화 로그 파일을 확인합니다.

+

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

+

- . 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다.
동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

[NOTE]

=====

{rhel} channels can be very large. Synchronization can sometimes take several hours.

=====

== GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.

[IMPORTANT]

=====

Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

=====

GPG 키에 대한 자세한 내용은 [xref:client-configuration:gpg-keys.adoc](#)[]에서 확인할 수 있습니다.

== 클라이언트 등록

To register your {redhat} clients, you need a bootstrap repository. By default, bootstrap repositories are automatically created, and regenerated daily for all synchronized products. You can manually create the bootstrap repository from the command prompt, using this command:

```
mgr-create-bootstrap-repo
```

클라이언트 등록에 대한 자세한 내용은 [xref:client-configuration:registration-overview.adoc](#)[]에서 참조하십시오.

```
:leveloffset: 3
:leveloffset: +2

[[rocky-registration-overview]]
= {rocky} 클라이언트 등록
```

{rocky} 클라이언트를 {productname} 서버에 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 다릅니다.

시작하기 전에 클라이언트의 날짜 및 시간이 {productname} 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [xref:client-configuration:activation-keys.adoc\[\]](#)를 참조하십시오.

```
:leveloffset: 3
:leveloffset: +3

[[clients-rocky]]
= {rocky} 클라이언트 등록
```

이 섹션에는 {rocky} 운영 체제를 실행하는 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

기존 클라이언트는 {rocky}에서 사용할 수 없습니다. {rocky} 클라이언트는 Salt 클라이언트로만 지원됩니다.

[NOTE]

=====

{rocky} 클라이언트를 {productname}에 등록하는 기능은 ``목표`` 정책으로 ``강제``하는 기본 SELinux 구성으로 테스트됩니다. SELinux를 비활성화해야 {rocky} 클라이언트를 {productname}에 등록할 수 있는 것은 아닙니다.

=====

== 소프트웨어 채널 추가

{rocky} 클라이언트를 {productname} 서버에 등록하려면 먼저 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

현재 지원되는 아키텍처는 ``x86_64`` 및 ``aarch64``이며, 버전 9의 경우 추가적으로 {ppc64le} 및 {s390x}가 지원됩니다. 지원되는 제품 및 아키텍처의 전체 목록은 [xref:client-configuration:supported-features.adoc\[\]](#)에서 확인할 수 있습니다.

[NOTE]

=====

다음 섹션에서 설명은 종종 [literal]``x86_64`` 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

====

이 절차에 필요한 채널은 다음과 같습니다.

```
[[rocky-channels-uyuni-cli]]
[cols="1,1,1,1", options="header"]
.{rocky} 채널 - CLI
| ===

| OS 버전 | 기본 채널 | 클라이언트 채널 | AppStream 채널
| {rocky} 9 | rockylinux9 | rockylinux9-uyuni-client | rockylinux9-appstream
| {rocky} 8 | rockylinux8 | rockylinux8-uyuni-client | rockylinux8-appstream
| ===

. 명령 프롬프트에서 소프트웨어 채널 추가
. {productname} 서버의 명령 프롬프트에서 root 권한으로 [command]``spacewalk-common-channels`` 명령을 사용해 적절한 채널을 추가합니다. 다음과 같이 올바른 아키텍처를 지정했는지 확인합니다.
+
```

spacewalk-common-channels \ -a <architecture> \ <base_channel_name> \ <child_channel_name_1> \ <child_channel_name_2> \ … <child_channel_name_n>

. xref:administration:custom-channels.adoc#_custom_channel_synchronization[automatic synchronization]이 꺼진 경우 다음 채널을 동기화하십시오.

+

spacewalk-repo-sync -p <base_channel_label>

. 계속하기 전에 동기화가 완료되었는지 확인합니다.

[NOTE]**=====**

[command]``spacewalk-common-channels``가 제공하는 클라이언트 도구 채널은 {suse}가 아닌 {uyuni}가 공급합니다.

=====**[IMPORTANT]****=====**

{rocky}{nbsp}8 및 {rocky}{nbsp}9 클라이언트의 경우 기본 채널과 AppStream 채널을 모두 추가하십시오. 사용자에게는 두 채널의 패키지가 모두 필요합니다. 두 채널을 모두 추가하지 않으면 패키지가 누락되어 부트스트랩 리포지토리를 생성할 수 없습니다.

=====**[NOTE]****=====**

AppStream 채널에서 사용할 수 있는 패키지의 수가 업스트림과 {productname} 채널 간에 서로 약간 불일치함을 알 수 있습니다. 또한 다른 시점에 추가한 동일 채널을 비교하면 그 수가 다른 것을 알 수 있습니다. 이는 {rocky}가 리포지토리를 관리하는 방식에 원인이 있습니다. {rocky}는 새 버전이 릴리스되면 이전 버전의 패키지를 제거하지만 {productname}은 사용 기간과 관계없이 모두 그대로 유지합니다.

=====

If you are using modular channels with {rocky} 8, you must enable the Python 3.6 module stream on the client. If you do not provide Python 3.6, the installation of the [package]``spacecmd`` package will fail.

[IMPORTANT]**=====**

AppStream 리포지토리는 모듈형 패키지를 제공합니다. 이로 인해 {productname} {webui}에 부정확한 패키지 정보가 표시됩니다. {webui} 또는 API를 사용해 모듈형 리포지토리에서 직접 설치하거나 업그레이드하는 등의 패키지 작업을 수행할 수 없습니다.

또는 Salt 상태를 사용해 Salt 클라이언트에서 모듈형 패키지를 관리하거나 클라이언트에서 [command]``dnf`` 명령을 사용할 수 있습니다. CLM에 대한 자세한 내용은 [xref:administration:content-lifecycle.adoc](#)[]을 참조하십시오.

=====

== 동기화 상태 확인

- . 절차: {webui}에서 동기화 진행률 확인
- . {productname} {webui}에서 menu:소프트웨어[관리 > 채널]로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
- . [guimenu] ``리포지토리`` 탭으로 이동한 다음, [guimenu] ``동기화``를 클릭하고 [systemitem] ``동기화 상태``를 확인합니다.

- . 절차: 명령 프롬프트에서 동기화 진행 상태 확인
- . {productname} 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 [command] ``tail`` 명령을 사용해 동기화 로그 파일을 확인합니다.

+

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

+

- . 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다.
- 동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

== 활성화 키 생성

{rocky} 채널에 연결된 활성화 키를 생성해야 합니다.

활성화 키에 대한 자세한 내용은 [xref:client-configuration:activation-keys.adoc](#)[]를 참조하십시오.

== GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.

[IMPORTANT]

=====

클라이언트의 보안을 위해 GPG 키의 신뢰성이 중요합니다. 어떤 키가 필요하고 신뢰할 수 있는지 결정하는 것은 관리자가 수행해야 하는 작업입니다. GPG 키를 신뢰할 수 없으면 클라이언트에 소프트웨어 채널을 할당할 수 없습니다.

=====

GPG 키에 대한 자세한 내용은 [xref:client-configuration:gpg-keys.adoc](#)[]에서 확인할 수 있습니다.

== 클라이언트 등록

{rocky} 클라이언트는 기타 모든 클라이언트와 동일한 방식으로 등록됩니다. 자세한 내용은 [xref:client-configuration:registration-overview.adoc](#)[]를 참조하십시오.

== 정오표 관리

{rocky} 클라이언트를 업데이트할 때 패키지에는 업데이트에 대한 메타데이터가 포함됩니다.

```
:leveloffset: 3  
:leveloffset: +2
```

[[ubuntu-registration-overview]]

= Ubuntu 클라이언트 등록

{ubuntu} 클라이언트를 {productname} 서버에 등록할 수 있습니다. 방법과 상세 정보는 클라이언트의 운영 체제에 따라 달라집니다.

시작하기 전에 클라이언트의 날짜 및 시간이 {productname} 서버와 정확히 동기화되었는지 확인하십시오.

또한 활성화 키 생성을 완료한 상태이어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [xref:client-configuration:activation-keys.adoc](#)[]를 참조하십시오.

```
:leveloffset: 3  
:leveloffset: +3
```

[[clients-ubuntu]]

= {ubuntu} 20.04 및 22.04 클라이언트 등록

이 섹션에는 {ubuntu} 20.04 LTS 및 22.04 LTS 운영 체제를 실행하는 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

Salt 클라이언트에 대해서만 {ubuntu}을 지원합니다. 기존 클라이언트는 지원하지 않습니다.

부트스트래핑은 {ubuntu} 클라이언트 시작과 초기 상태 실행 수행(예: 리포지토리 설정, 프로파일 업데이트 수행)에 대해 지원됩니다. 하지만 {ubuntu}의 루트 사용자는 기본적으로 비활성화되므로 부트스트래핑을 사용하려면 Python에 대한 [command]``sudo`` 권한이 있는 기존 사용자가 필요합니다.

[IMPORTANT]

=====

Canonical은 {productname}을 지지하거나 지원하지 않습니다.

=====

== 소프트웨어 채널 추가

{ubuntu} 클라이언트를 {productname} 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

[NOTE]

====

다음 섹션에서 설명은 종종 [literal] ``x86_64`` 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

====

이 절차에 필요한 채널은 다음과 같습니다.

```
[[ubuntu-channels-cli-uyuni]]
[cols="1,1,1,1,1,1", options="header"]
.Ubuntu 채널 - CLI
| ===
| OS 버전 | 기본 채널 | 주 채널 | 업데이트 채널 | 보안 채널 | 클라이언트 채널
| {ubuntu} 20.04 | ubuntu-2004-pool-amd64-uyuni | ubuntu-2004-amd64-main-uyuni | ubuntu-2004-amd64-main-updates-uyuni | ubuntu-2004-amd64-main-security-uyuni | ubuntu-2004-amd64-uyuni-client
| {ubuntu} 22.04 | ubuntu-2204-pool-amd64-uyuni | ubuntu-2204-amd64-main-uyuni | ubuntu-2204-amd64-main-updates-uyuni | ubuntu-2204-amd64-main-security-uyuni | ubuntu-2204-amd64-uyuni-client
| ===
```

Version 20.04 also requires the Universe channels:

```
[[ubuntu-universe-channels-cli-uyuni]]
[cols="1,1", options="header"]
.Ubuntu 20.04 Universe Channels - CLI
| ===
| {ubuntu} 20.04 | {nbsp}
| Universe Channel | ubuntu-2004-amd64-universe-uyuni
| Universe Updates Channel | ubuntu-2004-amd64-universe-updates-uyuni
| Universe Security Updates Channel | ubuntu-2004-amd64-universe-security-uyuni
```

```
| Universe Backports Channel | ubuntu-2004-amd64-universe-backports-uyuni
```

====

- . 명령 프롬프트에서 소프트웨어 채널 추가
- . {productname} 서버의 명령 프롬프트에서 root로 [command]``spacewalk-common-channels`` 명령을 사용하여 적절한 채널을 추가합니다.

+

```
spacewalk-common-channels \ <base_channel_label> \ <child_channel_label_1> \
<child_channel_label_2> \ ... <child_channel_label_n>
```

- . `xref:administration:custom-channels.adoc#custom_channel_synchronization[automatic synchronization]`이
꺼진 경우 다음 채널을 동기화하십시오.

+

```
spacewalk-repo-sync -p <base_channel_label>
```

- . 계속하기 전에 동기화가 완료되었는지 확인합니다.

[IMPORTANT]

====

새 채널을 완전히 동기화한 후에 Ubuntu 클라이언트를 부트스트랩해야 합니다.

====

== 동기화 상태 확인

- . 절차: {webui}에서 동기화 진행률 확인

. {productname} {webui}에서 menu:소프트웨어[관리 > 채널]로 이동한 다음, 리포지토리에
연결된 채널을 클릭합니다.

. [guimenu]``리포지토리`` 탭으로 이동한 다음, [guimenu]``동기화``를 클릭하고
[systemitem]``동기화 상태``를 확인합니다.

- . 절차: 명령 프롬프트에서 동기화 진행 상태 확인

. {productname} 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이
[command]``tail`` 명령을 사용해 동기화 로그 파일을 확인합니다.

+

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

+

- . 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다.
동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

[NOTE]

=====

{ubuntu} 채널은 규모가 매우 클 수 있습니다. 때로 동기화에 몇 시간이 걸릴 수 있습니다.

=====

== GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.

[IMPORTANT]

=====

Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

=====

GPG 키에 대한 자세한 내용은 [xref:client-configuration:gpg-keys.adoc](#)[]에서 확인할 수 있습니다.

== 루트 액세스

{ubuntu}의 루트 사용자는 SSH 액세스에 대해 기본적으로 비활성화되어 있습니다.

일반 사용자를 사용하여 온보딩하려면 [filename]``sudoers`` 파일을 편집해야 합니다.

- . 절차: 루트 사용자에게 액세스 권한 부여

- . 클라이언트에서 다음과 같이 [filename]``sudoers`` 파일을 편집합니다.

+

```
sudo visudo
```

+

이 줄을 [filename]``sudoers`` 파일 끝에 추가하여 사용자에게 [command]``sudo`` 액세스 권한을 부여합니다. 다음과 같이 [systemitem]``<user>``를 {webui}에서 클라이언트를 부트스트래핑하고 있는 사용자의 이름으로 교체합니다.

+

```
<user> ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2, /usr/bin/python3, /var/tmp/venv-salt-minion/bin/python
```

[NOTE]

=====

이 절차는 클라이언트 등록에 필요한 비밀번호가 없어도 루트에 액세스 권한을 부여합니다. 클라이언트는 성공적으로 설치된 후 루트 권한으로 실행되므로 액세스 권한이 더 이상 필요 없습니다. 클라이언트가 성공적으로 설치된 후 [path]``sudoers`` 파일에서 이 줄을 제거하는 것이 좋습니다.

=====

== 클라이언트 등록

{ubuntu} 클라이언트를 등록하려면 부트스트랩 리포지토리가 필요합니다. 기본적으로 부트스트랩 리포지토리는 자동으로 생성되며 동기화된 모든 제품에 대해 매일 재생성됩니다. 명령 프롬프트에서 다음 명령을 사용해 부트스트랩 리포지토리를 수동으로 생성할 수 있습니다.

```
mgr-create-bootstrap-repo
```

클라이언트 등록에 대한 자세한 내용은 [xref:client-configuration:registration-overview.adoc](#)[]에서 참조하십시오.

```
:leveloffset: 3  
:leveloffset: +3
```

```
[[clients-ubuntu-old]]  
= {ubuntu} 18.04 클라이언트 등록
```

이 섹션에는 {ubuntu} 18.04 LTS 운영 체제를 실행하는 Salt 클라이언트를 등록하는 방법에 대한 정보가 포함되어 있습니다.

{productname}은(는) Salt를 사용하는 18.04 LTS 클라이언트를 지원합니다.

Salt 클라이언트에 대해서만 {ubuntu}을 지원합니다. 기존 클라이언트는 지원하지 않습니다.

부트스트래핑은 {ubuntu} 클라이언트 시작과 초기 상태 실행 수행(예: 리포지토리 설정, 프로파일 업데이트 수행)에 대해 지원됩니다. 하지만 {ubuntu}의 루트 사용자는 기본적으로 비활성화되므로 부트스트래핑을 사용하려면 Python에 대한 [command]``sudo`` 권한이 있는 기존 사용자가 필요합니다.

[IMPORTANT]

=====

Canonical은 {productname}를 지지하거나 지원하지 않습니다.

=====

== 소프트웨어 채널 추가

{ubuntu} 클라이언트를 {productname} 서버에 등록하기 전에 필요한 소프트웨어 채널을 추가하고 이 채널을 동기화해야 합니다.

[NOTE]

=====

다음 섹션에서 설명은 종종 [literal]``x86_64`` 아키텍처로 기본 설정됩니다. 해당될 경우 다른 아키텍처로 교체하십시오.

=====

이 절차에 필요한 채널은 다음과 같습니다.

```
[[ubuntu-old-channels-cli-uyuni]]
[cols="1,1", options="header"]
```

```
.Ubuntu 채널 - CLI
| ====
| OS 버전 | {ubuntu} 18.04
| 기본 채널 | ubuntu-1804-pool-amd64-uyuni
| 주 채널 | ubuntu-1804-amd64-main-uyuni
| 업데이트 채널 | ubuntu-1804-amd64-main-updates-uyuni
| 보안 채널 | ubuntu-1804-amd64-main-security-uyuni
| 범용 채널1 | ubuntu-1804-amd64-universe-uyuni
| 범용 업데이트 채널 | ubuntu-1804-amd64-universe-updates-uyuni
| 범용 보안 업데이트 채널 | ubuntu-1804-amd64-universe-security-uyuni
| 클라이언트 채널 | ubuntu-1804-amd64-uyuni-client
```

```
| ====

```

```
//[NOTE]
//=====
//{ubuntu} 16.04 is now at end-of-life, and the ISO images provided in
the repository are out of date.
//Bootstrapping new {ubuntu} 16.04 clients using these packages will
fail.
//If you need to bootstrap new {ubuntu} 16.04 clients, follow the
troubleshooting procedure in xref:administration:troubleshooting/tshoot-
intro.adoc[Troubleshooting].
//=====
```

. 명령 프롬프트에서 소프트웨어 채널 추가
. {productname} 서버의 명령 프롬프트에서 root로 [command] ``spacewalk-common-
channels`` 명령을 사용하여 적절한 채널을 추가합니다.
+

```
spacewalk-common-channels \ <base_channel_label> \ <child_channel_label_1> \
<child_channel_label_2> \ ... <child_channel_label_n>
```

```
. xref:administration:custom-
channels.adoc#_custom_channel_synchronization[automatic synchronization]이
꺼진 경우 다음 채널을 동기화하십시오.
```

```
+
```

```
spacewalk-repo-sync -p <base_channel_label>
```

```
. 계속하기 전에 동기화가 완료되었는지 확인합니다.
```

[IMPORTANT]**=====**

Ubuntu 클라이언트를 부트스트래핑하기 전에 Universe(Universe에는 Salt에 대한 중요 종속성이 포함되어 있음)를 비롯한 모든 새 채널을 완전히 동기화해야 합니다.

=====**== 동기화 상태 확인****. 절차: {webui}에서 동기화 진행률 확인**

- . {productname} {webui}에서 menu:소프트웨어[관리 > 채널]로 이동한 다음, 리포지토리에 연결된 채널을 클릭합니다.
- . [guimenu]``리포지토리`` 탭으로 이동한 다음, [guimenu]``동기화``를 클릭하고 [systemitem]``동기화 상태``를 확인합니다.

. 절차: 명령 프롬프트에서 동기화 진행 상태 확인

- . {productname} 서버의 명령 프롬프트에서 루트 권한으로 다음과 같이 [command]``tail`` 명령을 사용해 동기화 로그 파일을 확인합니다.

+

```
tail -f /var/log/rhn/reposync/<channel-label>.log
```

+

- . 각 하위 채널은 동기화가 진행되는 동안 자체 로그를 생성합니다.
동기화가 완료되었는지 알아보려면 모든 기본 및 하위 채널 로그 파일을 확인해야 합니다.

[NOTE]**=====**

{ubuntu} 채널은 규모가 매우 클 수 있습니다. 때로 동기화에 몇 시간이 걸릴 수 있습니다.

=====

```
//ifeval:::{uyuni-content} == true]
```

```
//== Trust GPG Keys on Clients
```

```
//include::snippets/trust_gpg.adoc[ ]
```

```
//endif::[]
```

== GPG 키 관리

클라이언트는 소프트웨어 패키지가 설치되기 전에 GPG 키를 사용해 신뢰성을 확인합니다. 신뢰할 수 있는 소프트웨어만 클라이언트에 설치할 수 있습니다.

[IMPORTANT]

=====

Trusting a GPG key is important for security on clients. It is the task of the administrator to decide which keys are needed and can be trusted. Because a software channel cannot be used when the GPG key is not trusted, the decision of assigning a channel to a client depends on the decision of trusting the key.

=====

GPG 키에 대한 자세한 내용은 [xref:client-configuration:gpg-keys.adoc](#)[]에서 확인할 수 있습니다.

== 루트 액세스

{ubuntu}의 루트 사용자는 SSH 액세스에 대해 기본적으로 비활성화되어 있습니다.

일반 사용자를 사용하여 온보딩하려면 [filename]``sudoers`` 파일을 편집해야 합니다.

. 절차: 루트 사용자에게 액세스 권한 부여

- . 클라이언트에서 다음과 같이 [filename]``sudoers`` 파일을 편집합니다.

+
+

sudo visudo

+

이 줄을 [filename]``sudoers`` 파일 끝에 추가하여 사용자에게 [command]``sudo`` 액세스 권한을 부여합니다. 다음과 같이 [systemitem]``<user>``를 {webui}에서 클라이언트를 부트스트래핑하고 있는 사용자의 이름으로 교체합니다.

+

```
<user> ALL=NOPASSWD: /usr/bin/python, /usr/bin/python2, /usr/bin/python3, /var/tmp/venv-salt-minion/bin/python
```

[NOTE]

=====

이 절차는 클라이언트 등록에 필요한 비밀번호가 없어도 루트에 액세스 권한을 부여합니다. 클라이언트는 성공적으로 설치된 후 루트 권한으로 실행되므로 액세스 권한이 더 이상 필요 없습니다. 클라이언트가 성공적으로 설치된 후 [path]``sudoers`` 파일에서 이 줄을 제거하는 것이 좋습니다.

```
=====
```

== 클라이언트 등록

{ubuntu} 클라이언트를 등록하려면 부트스트랩 리포지토리가 필요합니다. 기본적으로 부트스트랩 리포지토리는 자동으로 생성되며 동기화된 모든 제품에 대해 매일 재생성됩니다. 명령 프롬프트에서 다음 명령을 사용해 부트스트랩 리포지토리를 수동으로 생성할 수 있습니다.

`mgr-create-bootstrap-repo`

클라이언트 등록에 대한 자세한 내용은 [xref:client-configuration:registration-overview.adoc](#)[]에서 참조하십시오.

:leveloffset: 3
:leveloffset: +2

[[client-proxy]]
= 클라이언트를 프록시에 등록

프록시 서버는 Salt 및 기존 클라이언트를 위한 브로커 및 패키지 캐시의 역할을 할 수 있습니다. 프록시에 클라이언트를 등록하는 작업은 몇 가지 주요 차이점 외에는 {productname} 서버에 직접 등록하는 것과 유사합니다.

이 섹션에는 {webui}, 명령줄의 명령 또는 부트스트랩 스크립트를 사용해 프록시에 Salt 클라이언트를 등록하는 내용이 포함되어 있습니다. 부트스트랩 스크립트를 사용해 기존 클라이언트를 등록하는 내용도 있습니다. 또한 {productname} 프록시에서 다른 프록시 또는 {productname} 서버로 클라이언트를 이동하는 절차도 있습니다.

{webui} 내에서 프록시 페이지에는 Salt 클라이언트와 기존 클라이언트에 관한 정보가 표시됩니다. 프록시에 연결된 클라이언트 목록은 menu:시스템[시스템 목록 > 프록시]에서 프록시 이름을 클릭한 다음, [guimenu]``상세 정보`` 탭의 [guimenu]``프록시`` 하위 탭을 선택하면 볼 수 있습니다.

Salt 클라이언트의 체인화된 프록시 목록은 menu:시스템[전체]에서 클라이언트 이름을 클릭한 다음 [guimenu]``상세 정보`` 탭의 [guimenu]``연결`` 하위 탭을 선택하면 볼 수 있습니다.

== 프록시 간 클라이언트 이동

등록 프로세스를 반복할 필요 없이 프록시 간에 Salt 및 Salt SSH 푸시 클라이언트를 이동할 수 있습니다.

[NOTE]

=====

프록시 간에 기존 클라이언트를 이동하려면 처음부터 등록 프로세스를 반복해야 합니다.

=====

. 절차: 프록시 간 Salt 또는 Salt SSH 푸시 클라이언트 이동

- . {productname} {webui}에서 프록시 간에 이동하려는 클라이언트의 [guimenu] `` 시스템 정보 `` 페이지로 이동합니다.
- . [guimenu] `` 연결 `` 탭으로 이동합니다. 그런 다음 [guimenu] `` 프록시 변경 `` 링크를 따라 드롭다운 메뉴를 확인합니다.
- . [guimenu] `` 새 프록시 `` 드롭다운 메뉴에서 클라이언트가 이동할 프록시를 선택하고 btn:[프록시 변경]을 클릭합니다.

. 절차: SSM으로 여러 프록시 간 Salt 또는 Salt SSH 푸시 클라이언트를 이동

- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 이동할 각 클라이언트의 확인란을 선택하면 클라이언트가 시스템 세트 관리자에 추가됩니다.
- . menu:시스템[시스템 세트 관리자]로 이동한 후 menu:기타[프록시] 탭으로 이동합니다.
- . [guimenu] `` 새 프록시 `` 드롭다운 메뉴에서 클라이언트를 이동할 프록시를 선택한 후 btn:[프록시 변경]을 클릭합니다.

[systemitem] `` system.changeProxy `` API 호출을 통해서도 동일한 기능을 이용할 수 있습니다.

==== 배경 정보

이 함수의 효과는 일반 Salt 클라이언트와 Salt SSH 푸시 클라이언트 간에 다릅니다.

==== 일반 Salt 클라이언트

함수는 Salt 상태 동작을 예약하며, 이를 통해 [path] `` susemanager.conf `` Salt 클라이언트 구성 파일의 [literal] `` master: `` 설정을 수정하여 새 프록시를 가리키도록 합니다. 이후에 함수는 Salt 클라이언트를 다시 시작합니다.

[NOTE]

=====

[path] `` susemanager.conf `` 파일을 수동으로 편집하여 [literal] `` master: `` 를 변경해도 효과가 같으며 이러한 기능도 지원됩니다.

=====

미니언이 다시 시작되고 새 프록시를 통해 다시 연결되면 서버는 데이터베이스의 프록시 경로를 업데이트하고 채널 URL을 새로 고치기 위한 다른 작업을 예약합니다.

==== Salt SSH 푸시 클라이언트

이 함수는 데이터베이스에서 즉시 프록시 경로를 업데이트하며 채널 URL을 새로 고치는 새로운 작업이 예약됩니다.

== 프록시에서 서버로 클라이언트 이동

Salt 클라이언트를 프록시에서 서버로 이동하려면 프록시 목록에서 [literal] ``없음``을 선택하십시오.

기존 클라이언트를 서버로 이동하려면 처음부터 등록 프로세스를 반복해야 합니다.

```
:leveloffset: 3
:leveloffset: +3
```

```
[[salt-client-proxy]]
= {webui}로 클라이언트를 프록시에 등록
```

{webui}를 사용하여 Salt 클라이언트를 {productname} 프록시에 등록할 수 있습니다.

A bootstrap repository is needed for non-SLE clients in general and for SLE clients before version 15. A bootstrap repository offers packages for installing Salt on clients and for registering Salt or traditional clients.

For information about creating a bootstrap repository, see [xref:client-configuration:bootstrap-repository.adoc](#)[].

. 절차: {webui}로 클라이언트를 프록시에 등록

- . {productname} {webui}에서 menu:시스템[부트스트랩]으로 이동합니다.
- . [guimenu] ``호스트`` 필드에 부트스트랩할 클라이언트의 정규화된 도메인 이름(FQDN)을 입력합니다.
- . [guimenu] ``SSH 포트`` 필드에서 클라이언트를 연결하고 부트스트랩하는 데 사용할 SSH 포트 번호를 입력합니다.
 - 기본적으로 SSH 포트는 [systemitem] ``22``입니다.
- . [guimenu] ``사용자`` 필드에서 클라이언트에 로그인할 사용자 이름을 입력합니다.
 - 기본적으로 사용자 이름은 [systemitem] ``root``입니다.
- . [guimenu] ``인증 방법`` 필드에서 클라이언트를 부트스트랩하는 데 사용할 인증 방법을

선택합니다.

+

- * 비밀번호 인증의 경우 [guimenu]``비밀번호`` 필드에서 클라이언트에 로그인할 때 사용할 비밀번호를 입력합니다.

- * SSH 개인 키 인증의 경우 개인 키와 개인 키에 연결된 비밀번호를 입력합니다.

이 키는 부트스트랩 프로세스를 완료하는 데 필요한 기간 동안만 보관됩니다.

- . [guimenu]``활성화 키`` 필드에서 클라이언트를 부트스트랩하는 데 사용할 소프트웨어 채널과 연결된 활성화 키를 선택합니다.

- . [guimenu]``프록시`` 필드에서 등록하려는 프록시 서버를 선택합니다.

- . 기본적으로 [guimenu]``SSH Strict 키 호스트 점검 비활성화`` 확인란이 선택되어 있습니다.

따라서 사용자가 수동으로 인증할 필요 없이 부트스트랩 프로세스가 자동으로 SSH 호스트 키를 수락할 수 있습니다.

- . 옵션: [guimenu]``SSH를 통해 시스템을 완전히 관리`` 확인란을 선택합니다.

이 옵션을 선택하면 서버에 연결하기 위해 SSH를 사용하도록 클라이언트가 구성되고 다른 연결 방법은 구성되지 않습니다.

- . btn:[부트스트랩]을 클릭하여 등록을 시작합니다.

부트스트랩 프로세스가 완료되면 클라이언트가 menu:시스템[시스템 목록]에 나열됩니다.

```
:leveloffset: 3
```

```
:leveloffset: +3
```

```
[[cli-client-proxy]]
```

```
== 명령줄에서 등록(Salt)
```

```
// Might need an 'unsupported' note? LKB 2019-05-01
```

```
// cf. https://bugzilla.suse.com/show_bug.cgi?id=1131398
```

```
// I'd say "no", according to the outcome of
```

```
// https://github.com/SUSE/spacewalk/issues/9333 KE 2019-12-17
```

{webui} 대신 명령줄을 사용해 Salt 클라이언트를 프록시에 등록할 수 있습니다. 이 절차를 진행하려면 등록하기 전에 먼저 Salt 클라이언트에 Salt 패키지를 설치해야 합니다. SLE 12 기반 클라이언트의 경우 [systemitem]``Advanced Systems Management`` 모듈도 활성화를 완료한 상태여야 합니다.

[NOTE]

=====

명령줄에서 기존 클라이언트를 등록할 수도 있지만 이 작업을 하려면 더 많은 단계를 거쳐야 합니다. 여기서는 이에 관해 다루지 않겠습니다. 부트스트랩 스크립트 절차를 사용해 기존 클라이언트를 등록하십시오. 자세한 내용은 xref:client-proxy-script.adoc[]를 참조하십시오.

=====

. 절차: 명령줄을 사용해 클라이언트를 프록시에 등록

- . 다음 위치에 있는 클라이언트 구성 파일을 선택합니다.

+
또는
+

/etc/salt/minion

+
또는
+

/etc/salt/minion.d/NAME.conf

+
이 파일을 종종 minion 파일이라고도 합니다.
. 다음과 같이 프록시 FQDN을 클라이언트 구성 파일에 `master`로 추가합니다.
+

master: PROXY123.EXAMPLE.COM

- . 다음과 같이 [systemitem]``salt-minion`` 서비스를 재시작합니다.

systemctl restart salt-minion

- . 서버에서 새로운 클라이언트 키를 수락하고, [systemitem]``<client>``를 클라이언트 이름으로 대체합니다.

salt-key -a '<client>'

```
:leveloffset: 3
:leveloffset: +3

[[script-client-proxy]]
= 부트스트랩 스크립트로 등록(Salt 및 기존)
```

부트스트랩 스크립트를 사용하여 {productname} 프록시를 통해 Salt 또는 기존 클라이언트를 등록할 수 있습니다. 이 작업은 {productname} 서버를 사용하여 직접 클라이언트를 등록하는 방법과 거의 동일합니다. 차이점은 명령줄 도구를 통해 {productname} 프록시에 부트스트랩

스크립트를 생성한다는 것입니다. 그런 다음, 부트스트랩 스크립트는 모든 필요한 정보를 클라이언트에 배포합니다. 부트스트랩 스크립트에는 활성화 키 또는 GPG 키와 같은 몇 가지 파라미터가 필요합니다. 이 파라미터는 특정 설정에 따라 달라집니다.

. 절차: 부트스트랩 스크립트로 클라이언트를 프록시에 등록

- . {webui}를 사용해 {productname} 서버에서 클라이언트 활성화 키를 생성합니다.

자세한 내용은 [xref:client-configuration:activation-keys.adoc\[\]](#)를 참조하십시오.

- . 프록시에서 [command]``mgr-bootstrap`` 명령줄 도구를 {rootuser}로 실행합니다.

필요한 경우 추가 명령줄 스위치를 사용해 부트스트랩 스크립트를 조정합니다. Salt 클라이언트 대신 기존 클라이언트를 설치하려면 [command]``--traditional`` 스위치를 사용해야 합니다.

+

사용 가능한 옵션을 보려면 명령줄에서 다음과 같이 [command]``mgr-bootstrap --help``를 입력하십시오.

+

`mgr-bootstrap --activation-keys=key-string`

+

- . 옵션: 출력된 부트스트랩 스크립트를 편집합니다.

- . 클라이언트에서 직접 또는 프록시에서 [command]``ssh``로 부트스트랩 스크립트를 실행합니다. 다음과 같이 [systemitem]``<bootstrap>``을 부트스트랩 스크립트 이름으로, [systemitem]``<client.example.com>``을 클라이언트의 호스트 이름으로 대체합니다.

+

`cat <bootstrap> | ssh root@<client.example.com> /bin/bash`

```
:leveloffset: 3
```

```
:leveloffset: +2
```

[[clients-pubcloud]]

= 공용 클라우드에서 클라우드 등록

{productname} 서버를 설정한 후에는 클라이언트 등록을 시작할 수 있습니다.

-- 제품 추가 및 리포지토리 동기화

클라이언트에 해당하는 제품을 이미 추가했고 리포지토리를 {productname}에 동기화했는지 확인하십시오. 클라이언트 등록에 사용되는 부트스트랩 리포지토리를 생성하려면 이 작업은

필수입니다.

자세한 내용은 [xref:installation-and-upgrade:pubcloud-setup.adoc#add-product-sync-repo](#)[]에서 확인할 수 있습니다.

== 온디맨드 이미지 준비

{suse}가 제공한 온디맨드 이미지에서 시작된 인스턴스는 자동으로 등록되고 업데이트 인프라 및 {sle} 모듈이 활성화됩니다. 온디맨드 이미지를 {productname} 클라이언트로 사용하려면 이 자동화를 비활성화한 후에 시작해야 합니다.

- . 절차: 온디맨드 이미지 준비
- . 온디맨드 인스턴스에 로그인합니다.
- . 명령 프롬프트에서 root로 등록 데이터 및 리포지토리를 제거합니다.

+

`registercloudguest --clean`

- . ```cloud-regionsrv-client``` 패키지를 제거합니다.

+

`zypper rm cloud-regionsrv-client`

- . 클라우드 공급자 관련 추가 패키지를 제거합니다.

+

* Amazon EC2 :

+

`zypper rm regionServiceClientConfigEC2 regionServiceCertsEC2`

- +
* Google Compute Engine:
+

`zypper rm cloud-regionsrv-client-plugin-gce regionServiceClientConfigGCE regionServiceCertsGCE`

- +
* Microsoft Azure:

+

```
zypper rm regionServiceClientConfigAzure regionServiceCertsAzure
```

{productname}을(를) {scc}에 등록하는 지침은 [xref:installation-and-upgrade:server-setup.adoc](#)[]에서 참조하십시오.

-- 클라이언트 등록

{productname} {webui}에서 menu:시스템[부트스트래핑]으로 이동하여 ``호스트``, ``SSH 포트``, ``사용자`` 및 ``암호`` 필드를 입력합니다. ``호스트`` 필드에서 안정적인 FQDN을 사용할 수 있는지 확인합니다. 그렇지 않으면 {productname}가 공용 클라우드에서 다른 일시적인 FQDNS를 제공하는 경우 호스트를 찾을 수 없습니다.

[NOTE]

=====

기존 클라이언트를 부트스트랩하려면 클라이언트에 로그인한 상태에서 서버의 호스트 이름을 확인할 수 있는지 확인하십시오. 서버의 FQDN을 클라이언트의 [path]``/etc/hosts``로컬 확인 파일에 추가해야 할 수도 있습니다. 서버의 로컬 IP 주소가 포함된 [command]``hostname -f`` 명령을 사용하여 확인할 수 있습니다.

=====

일반적으로 공용 클라우드 이미지는 사용자 이름 및 암호를 사용한 SSH 로그인을 허용하지 않고 인증서가 있는 SSH만 허용합니다. {webui}에서 부트스트랩하려면 사용자 이름 및 SSH 키를 사용하여 SSH 로그인을 활성화해야 합니다. 이 작업은 menu:시스템[부트스트랩]으로 이동한 후 권한 부여 방법을 변경하여 수행할 수 있습니다.

클라우드 공급자가 Microsoft Azure인 경우 사용자 이름 및 암호를 사용하여 로그인할 수 있습니다. 이렇게 하려면 AzureUser가 암호를 사용하지 않고 루트 권한으로 명령을 실행할 수 있도록 허용해야 합니다. 이 작업을 수행하려면 [path]``/etc/sudoers.d/waagent`` 파일을 열고 다음 줄을 추가하거나 편집합니다.

```
AzureUser ALL=(ALL) NOPASSWD: ALL
```

[WARNING]

=====

AzureUser가 암호를 사용하지 않고 루트 권한으로 명령을 실행할 수 있도록 허용하면 보안 위험이 발생합니다. 테스트 목적으로만 이 방법을 사용하십시오. 프로덕션 시스템에서는 이 기능을 사용하지 마십시오.

=====

부트스트랩 프로세스가 완료된 후에는 클라이언트가 menu:시스템[시스템 목록]에 나열됩니다.

- * 프로세스에 대한 더 많은 제어 권한을 원하거나 여러 클라이언트를 등록해야 하거나 기존 클라이언트를 등록하려면 부트스트랩 스크립트를 생성합니다. 자세한 설명은 [xref:client-configuration:registration-bootstrap.adoc](#)[]를 참조하십시오.
- * Salt 클라이언트와 프로세스에 대한 더 많은 제어 권한을 위해서는 명령줄에서 단일 명령을 실행하면 도움이 될 수 있습니다. 자세한 설명은 [xref:client-configuration:registration-cli.adoc](#)[]를 참조하십시오.
- * 공용 클라우드 이미지(예: AWS AMI)에서 실행된 클라이언트를 등록하는 경우에는 몇 가지 추가 구성을 수행하여 서로 덮어쓰지 않도록 해야 합니다. 복제 클라이언트 등록에 대한 자세한 내용은 [xref:administration:troubleshooting/tshoot-registerclones.adoc](#)[]에서 확인할 수 있습니다.

== 활성화 키

활성화 키는 기존 및 Salt 클라이언트에서 클라이언트가 올바른 소프트웨어 자격이 있는지, 적절한 채널에 연결하고 있는지, 관련 그룹에 가입되어 있는지 확인하는 데 사용됩니다. 각 활성화 키는 조직에 구속되며 키를 생성할 때 설정할 수 있습니다.

활성화 키에 대한 자세한 설명은 [xref:client-configuration:activation-keys.adoc](#)[]에서 참조하십시오.

```
:leveloffset: 3
:leveloffset: +3
```

[[automatic-client-registration]]
= Terraform에서 생성한 클라이언트의 자동 등록

Terraform에서 생성한 새 클라이언트는 {productname}에 자동으로 등록될 수 있습니다. 등록은 다음의 두 가지 방법으로 수행할 수 있습니다.

- * cloud-init 기반 등록
- * 원격 실행 제공자 기반 등록

[[cloud-init-based-client-registration]]
== cloud-init 기반 클라이언트 등록

cloud-init를 활용하여 등록하는 방법은 새로 생성된 가상 머신을 자동으로 등록하는 기본 방법입니다. 이 솔루션은 호스트에 대한 SSH 연결 구성을 방지합니다. 클라이언트 생성에 사용된 도구와 상관없이 사용할 수도 있습니다.

사용자는 {productname}에 머신을 자동으로 등록하기 위해 Terraform으로 이미지를 배포할 때 사용자 데이터 세트를 전달할 수 있습니다. [path]``user_data``는 부트스트랩에서 한 번만 실행되며 시스템이 처음 시작될 때만 실행됩니다.

cloud-init를 사용하여 클라이언트를 등록하기 전 사용자는 다음을 구성해야 합니다.

- * 부트스트랩 스크립트에 대한 자세한 설명은 [xref:client-configuration:registration-bootstrap.adoc](#)[]에서 참조하십시오.
- * 활성화 키에 대한 자세한 설명은 [xref:client-configuration:activation-keys.adoc](#)[]에서 참조하십시오.

이 명령은 부트스트랩 스크립트를 다운로드하고 새 머신이 생성될 때 등록하며, `cloud-init` 구성에 추가해야 합니다.

```
curl -s http://hub-server.tf.local/pub/bootstrap/bootstrap-default.sh | bash -s
```

[NOTE]

=====

프로비저닝을 변경하기 위해 [path]``user_data``가 업데이트될 때마다 Terraform은 시스템을 제거한 후 새 IP 등으로 다시 생성합니다.

=====

AWS의 `cloud-init`에 대한 자세한 내용은 다음을 참조하십시오.

.

https://registry.terraform.io/providers/hashicorp/template/latest/docs/data-sources/cloudinit_config

`cloud-init`의 예는 다음을 참조하십시오.

.

https://registry.terraform.io/providers/hashicorp/cloudinit/latest/docs/data-sources/cloudinit_config#example-usage

[[remote-exec-provisioner-based-client-registration]]

== `remote-exec` 구축 프로그램 기반 등록

Terraform의 `remote-exec` 구축 프로그램을 사용하여 새로 생성된 가상 머신을 자동으로 등록하기 위한 두 번째 솔루션입니다.

`remote-exec` 구축 프로그램은 새로 생성된 시스템과 상호 작용하여, SSH 연결을 열고 해당 시스템에서 명령을 실행할 수 있습니다.

[IMPORTANT]

=====

[literal]```remote-exec``` 구축 프로그램을 사용하여 클라이언트를 등록할 때 사용자는 Terraform을 실행하는 시스템이 생성 후 새 가상 시스템에 액세스할 수 있는지 확인해야 합니다.

=====

나머지 요구 사항은 <<`cloud-init-based-client-registration`>>을 사용할 때와

동일합니다.

- * 부트스트랩 스크립트에 대한 자세한 설명은 [xref:client-configuration:registration-bootstrap.adoc](#)[]에서 참조하십시오.
- * 활성화 키에 대한 자세한 설명은 [xref:client-configuration:activation-keys.adoc](#)[]에서 참조하십시오.

이 명령은 부트스트랩 스크립트를 다운로드하고 새 머신이 생성될 때 등록하며, 실행할 원격 명령으로 정의해야 합니다.

```
curl -s http://hub-server.tf.local/pub/bootstrap/bootstrap-default.sh | bash -s
```

`remote-exec` 제공자에 대한 자세한 내용은

<https://www.terraform.io/docs/provisioners/remote-exec.html>[]에서 참조하십시오.

`:leveloffset: 3`
`:leveloffset: +1`

`[[client-upgrades]]`
= 클라이언트 업그레이드

클라이언트는 자체 기본 운영 체제의 버전 관리 시스템을 사용하며 정기적인 업그레이드가 필요합니다.

{suse} 운영 체제를 사용하는 SCC 등록 클라이언트의 경우 `{productname}` `{webui}` 내에서 업그레이드를 수행할 수 있습니다. 지원되는 `{sle}{nbsp}15` 업그레이드 경로는 <https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-upgrade-paths.html>[]에서 참조하십시오.

SLE{nbsp}12를 실행하는 클라이언트를 SLE{nbsp}15로 업그레이드하는 경우 업그레이드가 자동화되지만 시작하기 전에 몇 가지 준비 단계를 수행해야 합니다. 자세한 내용은 [xref:client-configuration:client-upgrades-major.adoc](#)[]를 참조하십시오.

컨텐트 라이프사이클 관리자를 사용하여 클라이언트 업그레이드를 자동화할 수도 있습니다. 자세한 내용은 [xref:client-configuration:client-upgrades-lifecycle.adoc](#)[]을 참조하십시오.

서비스 팩 업그레이드 openSUSE Leap 마이너 버전 업그레이드 또는 `{sle}` 마이그레이션으로의 openSUSE Leap과 같은 제품 마이그레이션에 대한 자세한 내용은 [xref:client-configuration:client-upgrades-product-migration.adoc](#)[]을 참조하십시오.

등록되지 않은 openSUSE Leap 클라이언트를 업그레이드하는 방법에 대한 자세한 내용은 [xref:client-configuration:client-upgrades-uyuni.adoc](#)[]를 참조하십시오.

```
:leveloffset: 3
:leveloffset: +2

[[client-upgrades-major]]
= 클라이언트 - 주 버전 업그레이드
```

클라이언트에는 최신 업데이트가 모두 적용되고, 설치된 운영 체제용으로 사용 가능한 최신 서비스 팩(SP)이 있어야 합니다. 시작하기 전에 시스템이 최신 상태이고 모든 업데이트가 설치되었는지 확인하십시오.

업그레이드는 {yast} 및 {ay}에서 제어하며 Zypper는 사용하지 않습니다.

== 마이그레이션 준비

클라이언트를 SLE{nbsp}12에서 SLE{nbsp}15{nbsp}로 마이그레이션하려면 먼저 다음 작업을 수행해야 합니다.

- . 설치 미디어 준비
- . 자동 설치 가능한 배포판 생성
- . 활성화 키 생성
- . {ay} 프로파일 업로드

- . 절차: 설치 미디어 준비(예: SLE 15 SP2)
- . 다음과 같이 {productname} 서버에서 SLE{nbsp}15{nbsp}SP2 설치 미디어에 대해 로컬 디렉토리를 생성합니다.

+

```
mkdir -p /srv/images/sle15sp2
```

+

- . 설치 원본이 포함된 ISO 이미지를 다운로드하고 이 ISO 이미지를 서버에 마운트합니다.

+

```
mount -o loop DVD1.iso /mnt/
```

+

- . 마운트한 ISO의 모든 내용을 로컬 파일 시스템에 복사합니다.

+

```
cp -r /mnt/* /srv/images/sle15sp2
```

+

- . 복사가 완료되면 ISO 이미지를 마운트 해제합니다.

+

umount /mnt

+

[NOTE]

=====

이 이미지는 {unifiedinstaller}(으)로, 여러 개의 자동 설치 가능한 배포판에 사용할 수 있습니다.

=====

. 절차: 자동 설치 가능한 배포 생성

. {productname} {webui}에서 menu:시스템[자동 설치 > 배포판]으로 이동한 후 btn:[배포 생성]을 클릭합니다.

. [guimenu]``자동 설치 가능한 배포판 생성`` 섹션에서 다음 파라미터를 사용합니다.

* [guimenu]``배포판 레이블`` 섹션에 배포판의 고유 이름을 입력합니다.

글자, 숫자, 하이픈, 마침표, 밑줄만 사용하고, 이름을 구성하는 문자는 다섯 개 이상이어야 합니다. 예: ``sles15sp2-x86_64``.

* [guimenu]``트리 경로`` 필드에 설치 원본의 절대 경로를 입력합니다.

예: [path]``/srv/images/sle15sp2``.

* [guimenu]``기본 채널`` 필드에서 [systemitem]``x86_64용 SLE-Product-SLES15-SP2-Pool``을 선택합니다.

* [guimenu]``설치 프로그램 생성`` 필드에서 [systemitem]``SUSE Linux Enterprise 15``를 선택합니다.

* [guimenu]``커널 옵션`` 필드에는, 설치를 위해 부팅할 때 커널로 전달될 옵션을 입력합니다.

[option]``install=`` 파라미터와 [option]``self_update=0 pt.options=self_update`` 파라미터가 기본적으로 추가됩니다.

* [guimenu]``커널 후 옵션`` 섹션에는, 설치된 시스템을 처음 부팅할 때 커널로 전달될 옵션을 입력합니다.

. btn:[자동 설치 가능한 배포판 생성]을 클릭하여 저장합니다.

기존 SLE{nbsp}12{nbsp} 기본 채널에서 새로운 SLE{nbsp}15 채널로 전환하려면 활성화

키가 필요합니다.

- . 절차: 활성화 키 생성
 - . {productname} 서버 {webui}에서 menu:시스템[활성화 키]로 이동하여 [guimenu]``키 생성``을 클릭합니다.
 - . 키에 대한 설명을 입력합니다.
 - . 키를 입력하거나 공백으로 두어 자동 키를 생성합니다.
 - . 옵션: 사용을 제한하려면 [guimenu]``사용`` 텍스트 필드에 값을 입력합니다.
 - . [systemitem]``SLE-Product-SLES15-SP2-Pool for x86_64`` 기본 채널을 선택합니다.
 - . 옵션: [guimenu]``추가 시스템 유형``을 선택합니다.
자세한 내용은 [https://documentation.suse.com/sles/15-SP3/html/SLES-all/article-modules.html\[\]](https://documentation.suse.com/sles/15-SP3/html/SLES-all/article-modules.html[])을 참조하십시오.
 - . btn:[활성화 키 생성]을 클릭합니다.
 - . [guimenu]``하위 채널`` 탭을 클릭하고 필요한 채널을 선택합니다.
 - . btn:[키 업데이트]를 클릭합니다.

== 자동 설치 프로파일 생성

자동 설치 프로파일에는 시스템을 설치하는 데 필요한 모든 설치 및 구성 데이터가 포함되어 있습니다. 설치 완료 후 실행될 스크립트도 포함할 수 있습니다. 시작 지점으로 사용할 수 있는 예제 스크립트는 [https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST\[\]](https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST[])를 참조하십시오.

올바른 AutoYaST 업그레이드 설정은 다음을 참조하십시오.

link:[https://doc.opensuse.org/projects/autoyast/#CreateProfile-upgrade\[\]](https://doc.opensuse.org/projects/autoyast/#CreateProfile-upgrade[]).

- . 절차: 자동 설치 프로파일 생성
 - . {productname} {webui}에서 menu:시스템[자동 설치 > 프로파일]로 이동하여 자동 설치 프로파일 스크립트를 업로드합니다.
 - + 시작 지점으로 사용할 수 있는 예제 스크립트는 다음을 참조하십시오.
 - + <https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST>.

- . ``커널 옵션`` 필드에 ``autoupgrade=1``을 입력합니다.

+

선택 사항으로 ``Y2DEBUG=1`` 옵션을 포함해도 됩니다. 디버그 설정은 필수까지는 아니지만, 향후 발생 가능한 문제를 조사하는 데 유용하게 사용될 수 있습니다.

+

[IMPORTANT]

=====

Azure 클라우드에서 실행 중인 클라이언트는 ``커널 옵션``에 ``textmode=1 console=ttyS0``을 추가해야 합니다.

=====

- . 자동 설치 프로파일을 붙여 넣거나 파일 업로드 필드를 사용합니다.

- . **btn:[생성]**을 클릭하여 저장합니다.

- . 업로드한 프로파일에 변수를 설정해야 하는 경우 **menu:시스템[자동 설치 > 프로파일]**로 이동하여 편집할 프로파일을 선택하고 **[guimenu]``변수``** 탭으로 이동합니다.

+

다음 형식을 사용하여 필수 변수를 지정합니다.

+

<key>=<value>

== 마이그레이션

시작하기 전, 자동 설치 프로파일에서 참조된 모든 채널을 사용할 수 있고 완전히 동기화되었는지 확인합니다.

[path]``/var/log/rhn/reposync/<channel-label>.log``에서 미러링 프로세스를 모니터링할 수 있습니다.

. 절차: 마이그레이션

. **{productname}** 서버 **{webui}**에서 **[guimenu]``시스템``**으로 이동하여 업그레이드할 클라이언트를 선택합니다.

. **[guimenu]``조달``** 탭으로 이동하여 업로드한 자동 설치 프로파일을 선택합니다.

. **btn:[자동 설치 일정 잡기 후 완료]**를 클릭합니다. 시스템이 필요한 파일을 다운로드하고, 부트로더 항목을 변경하고, 재부팅한 후 업그레이드를 시작합니다.

클라이언트는 다음번에 **{productname}** 서버와 동기화될 때 재설치 작업을 수행합니다. 재설치 작업은 새로운 커널 및 **initrd** 패키지를 가져옵니다. 또한 새 커널 및 **initrd** 패키지에 대한

포인터가 포함된 새로운 [path]``/boot/grub/menu.lst`` (GRUB Legacy) or [path]``/boot/grub2/grub.cfg`` (GRUB 2)를 작성합니다.

클라이언트는 다음번 부팅 시 grub을 사용해 initrd로 새 커널을 부팅합니다. 이 프로세스 중에는 PXE 부팅이 사용되지 않습니다.

작업을 가져온 후 약 3분이 지나면 클라이언트는 재부팅을 시작합니다.

[NOTE]

=====

Salt 클라이언트의 경우 마이그레이션이 완료되면 ``spacewalk/minion_script`` 코드 조각을 사용해 클라이언트를 다시 등록합니다.

=====

```
:leveloffset: 3
:leveloffset: +2
```

[[client-upgrades-clm]]

= 컨텐트 라이프싸이클 관리자를 사용하여 업그레이드

관리할 {sles} 클라이언트가 많은 경우 컨텐트 라이프싸이클 관리자를 사용하여 현재 위치에 업그레이드를 자동화할 수 있습니다.

== 업그레이드 준비

클라이언트를 업그레이드하려면 먼저 다음과 같은 준비 작업을 마쳐야 합니다.

- * 새 컨텐트 라이프싸이클 프로젝트 만들기
- * 활성화 키 생성
- * 자동 설치 가능한 배포판 생성
- * 자동 설치 프로파일 생성

. 절차: 컨텐트 라이프싸이클 프로젝트 생성

. 배포를 위한 컨텐트 라이프싸이클 프로젝트를 생성합니다.

+

자세한 내용은 [xref:administration:content-lifecycle.adoc](#)[]을 참조하십시오.

- . 프로젝트에 대해 설명하는 짧은 이름을 선택해야 합니다.
- . 배포에 필요한 모든 소스 채널 모듈을 포함합니다.
- . 필요에 따라 필터를 추가하고 하나 이상의 환경을 설정합니다.

. 절차: 활성화 키 생성

- . 배포를 위한 활성화 키를 생성합니다.

+

자세한 내용은 [xref:client-configuration:activation-keys.adoc\[\]](#)를 참조하십시오.

- . 활성화 키에 필터링된 프로젝트 채널이 모두 포함되어 있어야 합니다.

. 절차: 자동 설치 가능한 배포 생성

- . 마이그레이션하려는 모든 기본 채널에 대해 자동 설치 가능한 배포판을 생성합니다.

+

자세한 설명은 [xref:client-configuration:autoinst-distributions.adoc\[\]](#)를 참조하십시오.

- . 컨텐트 라이프싸이클 프로젝트의 이름을 참조하는 레이블을 배포에 지정합니다.

- . ``설치 프로그램 생성`` 필드에서 사용 중인 SLES 버전을 선택합니다.

. 절차: 자동 설치 프로파일 생성

- . 업그레이드하여 도달하려는 모든 대상 배포 및 서비스 팩에 대해 자동 설치 프로파일을 생성합니다.

+

자세한 설명은 [xref:client-configuration:autoinst-profiles.adoc\[\]](#)를 참조하십시오.

- . Salt 및 기존 클라이언트에 대해 다른 프로파일을 생성해야 합니다.

- . 프로파일의 변수를 사용해 다양한 라이프싸이클 환경을 구별할 수 있습니다.

자동 설치 프로파일의 예는 [https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST\[\]](https://github.com/SUSE/manager-build-profiles/tree/master/AutoYaST[])를 참조하십시오.

현재 위치에 업그레이드를 자동화하기 위해 자동 설치 프로파일에서 다음 변수를 사용하십시오.

. 예: 자동 설치 프로파일에서 사용할 변수

```
registration_key=1-15sp1-demo-test org=1 channel_prefix=15sp1-demo-test distro_label=15sp1-demo-test
```

. 예: 자동 설치 프로파일에서 사용할 항목

```

<listentry>
    <ask_on_error config:type="boolean">true</ask_on_error>

    <media_url>https://$redhat_management_server/ks/dist/child/$channel_prefix-sle-module-web-scripting15-sp1-pool-x86_64/$distro_label</media_url>
        <name>$channel_prefix SLE-Module-Web-Scripting15-SP1 Pool for x86_64
    </name>
        <product>Web Scripting Module 15 SP1 x86_64 Pool</product>
    </listentry>

```

== 업그레이드

업그레이드를 위해 서버 준비를 완료했으면 이제 클라이언트를 조달할 수 있습니다.

. 절차: 클라이언트 조달

- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 조달할 클라이언트를 선택하고 선택한 클라이언트에 시스템 세트 관리자를 추가합니다.
- . menu:시스템[시스템 세트 관리자 > 개요]로 이동하여 [guimenu]``조달`` 탭을 클릭합니다.
- . 사용할 자동 설치 프로파일을 선택합니다.

PXE를 사용할 수 있는 클라이언트의 경우 마이그레이션은 클라이언트를 조달하자마자 자동화됩니다. 기타 모든 클라이언트의 경우에는 Cobbler를 사용해 업그레이드를 수행할 수 있습니다.

. 절차: Cobbler를 사용한 클라이언트 업그레이드

- . 다음과 같이 명령 프롬프트에서 루트 권한으로 사용 가능한 Cobbler 프로파일을 확인합니다.
- +

cobbler 프로파일 목록

+

- . 다음과 같이 선택한 프로파일 및 배포로 ISO 파일을 빌드합니다.
- +

cobbler buildiso --iso=/tmp/SLE_15-sp1.iso --profiles=SLE_15-sp1:1:Example --distro=SLE_15-sp1

+

CD-ROM을 사용하여 클라이언트를 프로비저닝하는 방법은 [xref:client-configuration:autoinst-cdrom.adoc\[\]](#)를 참조하십시오.

```
:leveloffset: 3
:leveloffset: +2
```

[[client-upgrades-spmigration]]

= 제품 마이그레이션

Product migration allows you to upgrade SLE-based client systems from an Service Pack (SP) level to a later one. For example, you can migrate {sles} 15 SP1 to {sles} 15 SP2.

You can also migrate openSUSE Leap to a later minor version or to the corresponding {sles} SP level, for example:

- * openSUSE Leap 15.1에서 15.2로 또는
- * openSUSE Leap 15.1 to {sles} 15 SP1, or
- * openSUSE Leap 15.5 to {sles} 15 SP5

[WARNING]

====

마이그레이션 과정에서 {productname}는 설치 전에 필요한 모든 라이선스(EULA)를 자동으로 수락합니다.

====

In {sles} 12 and later, {suse} supports service pack skipping if {scc} provides it. For example, you can upgrade from {sles} 15 to SP2, without installing SP1. For supported {sles} upgrade paths, see <https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-upgrade-paths.html#sec-upgrade-paths-supported>.

[IMPORTANT]

====

Product migration is for upgrading within the same major version. You cannot use product migration to migrate from {sles} 12 to {sles} 15. For more information about major upgrades, see [xref:client-configuration:client-upgrades-major.adoc\[\]](#).

====

[WARNING]

====

제품 마이그레이션에는 롤백 기능이 없습니다. 마이그레이션 절차가 시작되면 롤백이 불가능합니다. 비상 상황에서 사용할 수 있는 시스템 백업이 있는지 확인하십시오.

=====

== 제품 마이그레이션 수행

제품 마이그레이션을 시작하기 전에:

- * Ensure there are no pending updates or patches. Check the [guimenu] ``System Status`` on the client system's menu:Details[Overview] page, and install all offered updates or patches. If your client system is not up-to-date, product migration may fail.
- * 대상 제품의 모든 채널이 완전히 동기화되었는지 확인하십시오. {webui}에서 동기화 상태를 확인하려면 menu:Admin[설정 마법사 > 제품] 페이지로 이동하십시오.

. 절차: 제품 마이그레이션 수행

- . menu:시스템[개요] 페이지에서 클라이언트를 선택합니다.
- . 클라이언트의 시스템 정보 페이지에서 menu:소프트웨어[제품 마이그레이션] 탭으로 이동합니다.
- . 대상 마이그레이션 경로를 선택하고 btn:[채널 선택]을 클릭합니다.
- . [guimenu] ``제품 마이그레이션 - 채널`` 페이지에서 ``필수 하위 채널``과 추가 ``옵션 하위 채널``을 포함하여 올바른 기본 채널을 선택합니다.
- . 선택 사항: [guimenu] ``벤더 변경 허용``을 선택하여 벤더가 변경된 패키지를 설치할 수 있도록 허용합니다. 이 경우 마이그레이션이 시작되기 전에 세부 정보와 함께 알림이 표시됩니다.

+

[IMPORTANT]

=====

To migrate openSUSE Leap to {sles}, you must check the [guimenu] ``Allow Vendor Change`` option.

=====

- . 채널이 적절히 구성되었으면 btn:[마이그레이션 예약]을 클릭합니다.

== 제품 대량 마이그레이션

다수의 클라이언트를 다음 SP 버전으로 마이그레이션하고 싶은 경우 {productname} API 호출을 사용하면 됩니다.

[WARNING]

=====

제품 대량 마이그레이션 작업은 위험합니다. 실수로 시스템을 업그레이드하지 않도록 주의하십시오. 프로세스를 철저히 테스트해야 합니다. 최소한으로 시험 실행을 우선 수행하십시오.

```
====
```

[command] ``spacecmd`` 명령줄 도구는

[systemitem] ``system_scheduleproductmigration`` 하위 명령을 제공하며, 이 하위 명령을 사용하여 많은 수의 클라이언트에 대해 다음 부 버전으로의 마이그레이션을 예약할 수 있습니다.

[systemitem] ``system_scheduleproductmigration``에 대한 구문 사용 및 옵션을 보려면 다음을 실행하십시오.

spacecmd system_scheduleproductmigration help

```
==== 제품 대량 마이그레이션 수행
```

- . 절차: 제품 대량 마이그레이션 수행

- . 다음과 같이 사용 가능한 마이그레이션 대상을 나열하고, 마이그레이션하려는 시스템 ID를 적어둡니다.

+

spacecmd api — system.listMigrationTargets -A 1000010001

- . 각 시스템 ID에 대해 [systemitem] ``listMigrationTarget`` 을 호출하고 원하는 대상 제품이 사용 가능한지 확인합니다.

+

- * 시스템 ID에 사용 가능한 대상이 있는 경우 [systemitem] ``system.scheduleProductMigration`` 을 호출합니다.

- * 원하는 대상을 사용할 수 없는 경우 시스템을 건너뛰십시오.

다음 템플릿을 환경에 맞게 조정하십시오.

```
target = '[...]' basechannel = 'channel-label' system_ids = [1, 2, 3]
```

```
session = auth.login(user, pass) for system in system_ids if system.listMigrationTargets(session, system).ident == target system.scheduleProductMigration(session, system, target, basechannel, [], False, <now>) else print "요청된 대상으로 마이그레이션할 수 없음 — 시스템 건너뛰기" endif endfor
```

```
==== 제품 대량 마이그레이션 예: SLES 15 SP2에서 SLES 15 SP3로
```

```
////
```

```
Adjust the following draft text
```

```
////
```

이 예에서는 대량 마이그레이션을 편리하게 수행할 수 있도록 그룹이 임시로 생성됩니다.

. 절차: 대량 제품 마이그레이션 그룹 생성

- . {productname} {webui}에서 menu:시스템[시스템 그룹]으로 이동하여 btn:[그룹 생성]을 클릭합니다.
- . [literal]``mpm-target-sles15sp3`` 그룹의 이름을 지정하십시오.

동일한 기본 채널에 등록된 시스템만 생성된 그룹에 추가해야 합니다. 이 예에서는 [literal]``SLE-Product-SLES15-SP2-Pool for x86_64``에 등록된 시스템만 그룹에 추가되었습니다.

[NOTE]

=====

지금 업그레이드하지 않을 시스템은 그룹에서 제거해야 합니다.

=====

. 절차: 시스템을 그룹에 추가

- . 그룹에 클라이언트를 추가하는 방법에 대한 자세한 내용은 xref:client-configuration:system-groups.adoc#_add_clients_to_groups[]에서 참조하십시오.

////

Note or warning about taking normal precautions (backups, make sure fully patched, etc)

////

[IMPORTANT]

=====

[command]``spacecmd`` 하위 명령

[literal]``system_scheduleproductmigration`` 및

[command]``system_listmigrationtargets``는 그룹의 일부에 해당하는 모든 시스템을 반복합니다. 그룹에 100개의 시스템이 있는 경우 100개의 작업이 예약됨으로 표시됩니다. 그룹의 모든 시스템이 동일한 "마이그레이션 대상"을 지원하는 것이 중요합니다.

그룹 내의 모든 시스템에 대한 대상을 가져오려면 다음을 실행:

```
spacecmd --system_listmigrationtargets group:mpm-target-sles15sp3
```

모든 시스템에 대해 보고되는 대상만 선택하십시오. 이 명령은 "IDs" 문자열을 출력합니다. 문자열은 다른 명령의 [literal]``MIGRATIONTARGET``에 대한 식별자입니다.

=====

. 절차: 대량 마이그레이션 명령 실행

- . 이 예에서 [literal]``mpm-target-sles15sp3`` 그룹의 모든 시스템을 SLES 12 SP2에서 SLES 15 SP로 업그레이드하려면 명령줄에 다음을 입력:
 - +

```
spacecmd --system_scheduleproductmigration group:mpm-target-sles15sp3 \ sle-product-sles15-sp3-pool-x86_64 "[190,203,195,1242]" -d
```

- +
- . [command] ``system_scheduleproductmigration`` 명령의 구문은 다음과 같습니다.
 - +

```
spacecmd --system_scheduleproductmigration <SYSTEM> <BASE_CHANNEL_LABEL> \
<MIGRATION_TARGET> [옵션]
```

자세한 내용을 보려면 [command] ``spacecmd -- system_scheduleproductmigration help`` 명령을 참조하십시오.

==== 필수 구문

<시스템>::

이 예에서는 해당 그룹에서 모든 시스템을 선택하기 위해 생성한 그룹을 사용합니다.

- +

group:mpm-target-sles15sp3

<BASE_CHANNEL_LABEL>::

대상 기본 채널의 레이블입니다. 이 경우 시스템은 SLES 15 SP3로 업그레이드되고 레이블은 [literal]``sle-product-sles15-sp3-pool-x86_64``입니다.

- +

현재 미러링되는 모든 기본 채널 목록을 보려면 다음을 실행:

- +

spacecmd softwarechannel_listbasechannels.

- +

현재 기본 채널에 대해 사용할 수 있는 대상이 아닌 경우 채널로 업그레이드할 수 없음에 유의하십시오.

<MIGRATION_TARGET>::

[literal]``group:mpm-target-sles15sp3`` 그룹의 시스템에 대해 이 값을 식별하려면 다음을 실행:

+

```
spacecmd --system_listmigrationtargets group:mpm-target-sles15sp3
```

+

[literal]``MIGRATION_TARGET`` 매개 변수가 전달되어야 하는 형식이며, 괄호 사용으로 인한 부작용을 방지하기 위해 셸 인용구 필요:

+

```
"[190,203,195,1242]"
```

옵션::

+

- . -s START_TIME
- . -d 시험 실행을 수행하려면 경우 이 플래그를 전달합니다(실제 마이그레이션을 수행하기 전 시험 실행을 실행하는 것이 권장됨).
- . -c CHILD_CHANNELS(쉼표로 구분된 하위 채널 레이블(공백 없음))

+

이 경우 성공적인 시험 실행 후 제거할 수 있는 [literal]``-d`` 옵션이 포함되었습니다.

성공하면 예약된 각 시스템에 대해 다음과 같은 명령 출력이 표시됩니다.

- . 시스템 mpm-sles152-1에 대한 제품 마이그레이션 예약
- . 예약된 작업 ID: 66

그룹의 지정된 시스템에 대해 {webui}에서 작업, 이 경우 시험 실행을 추적할 수도 있습니다. 클라이언트의 시스템 세부 정보 페이지에서 menu:Events[이력]으로 이동하십시오. 시험 실행 중 오류가 발생하면 시스템을 조사해야 합니다.

모두 정상이면 명령에서 [literal]``-d`` 옵션을 제거하여 실제 마이그레이션을 실행할 수 있습니다. 마이그레이션이 완료되면 {productname} {webui}에서 시스템을 재부팅할 수 있습니다.

```
:leveloffset: 3
```

```
:leveloffset: +2
```

```
[[client-upgrades-uyuni]]
= Uyuni 클라이언트 업그레이드
```

이 섹션에서는 openSUSE Leap을 예로 사용합니다.

== 업그레이드 준비

- . 절차: 클라이언트 업그레이드 준비
- . {productname} 서버의 명령 프롬프트에서 루트 권한으로 [command]``spacewalk-common-channels`` 명령을 사용해 적절한 채널을 추가합니다.
- +

```
spacewalk-common-channels  \  opensuse_leap15_4  \  opensuse_leap15_4-non-oss  \
opensuse_leap15_4-non-oss-updates\opensuse_leap15_4-updates\opensuse_leap15_4-uyuni-client
```

- . [command]``spacewalk-repo-sync``로 모든 채널을 완전히 동기화합니다. 이미 정의된 리포지토리 URL인 경우 xref:installation-and-upgrade:proxy-uyuni.adoc#uyuni-202007-channeldupes[]로 계속합니다.
- +

// These are custom channels.

// For more information, see xref:client-configuration:clients-opensuse.adoc[].

- . {productname} 서버 {webui}에서 menu:소프트웨어[관리 > 채널]로 이동하여 이름이 [systemitem]``Uyuni Client Tools for openSUSE Leap 15.4(x86_64)``인 채널을 클릭합니다.

- . 오른쪽 상단에서 btn:[채널 관리]를 클릭합니다.

- . [guimenu]``리포지토리`` 탭을 클릭하고 [systemitem]``외부 - openSUSE Leap 15.3(x86_64)``용 Uyuni 클라이언트 도구``를 선택합니다.

- . btn:[리포지토리 업데이트]를 클릭합니다.

- . menu:리포지토리[동기화] 하위 탭으로 이동하여 btn:[지금 동기화]를 클릭합니다.

- . [systemitem]``openSUSE Leap 15.4(x86_64)`` 및 [systemitem]``외부 - openSUSE Leap 15.3(x86_64)``에 대해서도 동일한 작업을 합니다.

[systemitem]``openSUSE Leap 15.4(x86_64)``를 펼쳐 패키지로 채워져 있는 모든 하위 채널을 표시합니다.

== 업그레이드

클라이언트를 업그레이드하려면 소프트웨어 리포지토리를 교체하고 소프트웨어를 업데이트한 후 끝으로 클라이언트를 재부팅하십시오.

- . 절차: 클라이언트 업그레이드

- . {productname} 서버 {webui}에서 menu:시스템[]으로 이동하여 클라이언트의 이름을 클릭합니다.
 - . menu:소프트웨어[소프트웨어 채널]을 클릭하고 [systemitem]``Customs Channels`` 목록에 나열된 openSUSE Leap {opensuse-version} 채널을 기본 채널로 선택합니다.
 - . [guimenu]``하위 채널`` 창에서 {opensuse-version} 하위 채널을 선택합니다.
 - . btn:[다음]을 클릭한 후 [guimenu]``소프트웨어 채널 변경 확인``을 클릭하여 btn:[확인]합니다.
 - . menu:소프트웨어[패키지 > 업그레이드]를 클릭하고 클라이언트에서 업데이트할 패키지를 모두 선택한 다음, 선택을 적용합니다. btn:[패키지 업그레이드]를 클릭하고, 상세 정보를 확인한 후 btn:[확인]을 클릭하여 업데이트를 완료합니다.
- +
- ```
// . Re-register with the reactivation key using the
[command]``rhnreg_ks`` command-line utility.
```
- +
- ```
// The system is re-registered with the same id, history, and groups.
```
- +
- ```
// and channels (unless the system's base channel changes).
```
- . 클라이언트를 재부팅합니다.

여러 클라이언트를 업데이트해야 하는 경우 {productname} 서버에서 이 명령 시퀀스의 작업 체인을 생성하면 됩니다. 이 작업 체인을 사용해 여러 클라이언트에서 업데이트를 동시에 수행할 수 있습니다.

- ```
////
```
- . Assign the new channels to the clients instead of the old channels.
 - . Update all the packages. This can either be done with the {webui} or better run [command]``zypper dup`` manually on the command line local on the systems or remotely as a Salt command.
- ```
////
```

```
////
```

I think the better way to document this is if giving it a try. Create an Uyuni server, sync Leap 15.1 (spacewalk-common-channels), create a Leap 15.1, onboard it, sync Leap 15.2 (spacewalk-common-channels), and then try to adjust the channels and trying to upgrade. I recommend you use VMs and take snapshots of the VMs so you can repeat steps as needed.

```
////
```

```
////
```

But for now you need to create and mirror at least the target channels with spacewalk-common-channels.

You adjust the channels for the client and best is to call "zypper dup".

Either from the commandline on that system or using remote command.

////

:leveloffset: 3

:leveloffset: +1

[ [delete.clients]]

= 클라이언트 삭제

// FIXME: where do we need to add warnings (suse clients only, all clients)

If you need to remove a client from your {productname} Server, you can use the {webui} to delete it. You can also remove a client from the command line. These procedures work for both traditional and Salt clients.

// can also be done manually.

// FIXME: Why Manual Cleanup is necessary sometimes.

[ [delete.clients.webui]]

-- Delete a Client with the {webui}

. 절차: 클라이언트 삭제

. {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 삭제하려는 클라이언트를 선택합니다.

. btn:[시스템 삭제]를 클릭합니다.

. 상세 정보를 확인하고 btn:[프로파일 삭제]를 클릭하여 확인합니다.

. Salt 클라이언트의 경우 {productname}은 추가 구성을 정리하려 합니다. 클라이언트에 연결할 수 없는 경우 삭제를 취소하거나 구성 파일을 정리하지 않고 클라이언트를 삭제할 수 있는 옵션이 제공됩니다.

시스템 세트 관리자를 사용해 여러 클라이언트를 삭제할 수도 있습니다. 시스템 세트 관리자에 대한 자세한 내용은 [xref:client-configuration:system-set-manager.adoc](#)[]를 참조하십시오.

[IMPORTANT]

=====

It is not possible to automatically clean up a traditional client after deleting it. You have to take care of this yourself. Furthermore, cleaning up a Salt client does not remove Salt itself.

=====

[NOTE]

```
====
```

Normally you migrate a traditional client to a Salt client without deleting the client. Salt automatically detects that you have a traditional client and does the necessary changes itself. But if you already deleted the traditional client and want to register it as a Salt client again, see [xref:administration:troubleshooting/tshoot-register-trad-as-salt-after-deletion.adoc](#)[].

```
====
```

```
-- Delete a Salt Client on the Command Line (with API Call)
```

.Procedure: Deleting a Client from the Server

. Delete the client with the FQDN (Fully Qualified Domain Name):

```
+
```

`spacecmd system_delete FQDN`

```
+
```

`[command] ``spacecmd system_delete`` also deletes the Salt key.`

`[command] ``system_delete`` offers the following options:`

`usage: system_delete [options] <SYSTEMS>`

`options:`

- `-c TYPE - Possible values:`
  - \* 'FAIL\_ON\_CLEANUP\_ERR' - fail in case of cleanup error,
  - \* 'NO\_CLEANUP' - do not cleanup, just delete,
  - \* 'FORCE\_DELETE' - try cleanup first but delete server anyway in case of error

```
////
```

`// move to Trouble Shooting and link from here`

Sometimes a new registration of a deleted (unregistered) client might not be possible.

To solve this issue, some Salt cache files should be deleted on the {productname} Server (Salt master) before trying to re-register again:

`rm /var/cache/salt/master/thin/version rm /var/cache/salt/master/thin/thin.tgz`

```
////
```

```
[[delete.clients.commandline]]
== Delete a Client from the Command Line

==== Salt Client

// Manual Registration Cleanup

This process is only for {productname} clients, do not run it on the
{productname} Server itself.

[NOTE]
=====

You must not execute the following procedure on clients running {rhel},
{debian}, or clones. Instead of [command]``zypper`` use equivalent
packager commands such as [command]``yum``, [command]``dnf``, or
[command]``apt``.

=====

.Procedure: Deleting SLES 12 and 15 Salt Clients

. Stop the salt-minion service:
+
```

systemctl stop salt-minion

```
+
/////
SLES 11:
+
```

rcsalt stop

```
////
. Remove the repositories and configuration files:
+
```

rm /etc/zypp/repos.d/susemanager\\*:channels.repo rm -r /etc/sysconfig/rhn/ rm -r /etc/salt/

- . Remove Client Packages:
- +

```
zypper rm salt salt-minion python*-salt sle-manager-tools-release
```

- .Procedure: Salt Bundle Client - Manual Registration Cleanup
- . To unregister, run:
- +

```
systemctl stop venv-salt-minion zypper rm -y venv-salt-minion rm /etc/zypp/repos.d/susemanager\channels.repo /etc/venv-salt-minion/* rm -r /etc/venv-salt-minion/*
```

For information about the Salt bundle, see [xref:client-configuration:contact-methods-saltbundle.adoc](#)[ ].

This process is only for {productname} clients, do not run it on the {productname} Server itself.

[NOTE]

====

You must not execute the following procedure literally on clients running {rhel}, {debian}, or clones. Instead of [command]``zypper`` use equivalent packager commands such as [command]``yum``, [command]``dnf``, or [command]``apt``.

====

.Procedure: Traditional SLES 12 and 15 Clients - Manual Cleanup

- . Stop the osad service (if it is in use):
- +

```
systemctl stop osad
```

```
// SLES 11:
// rcosad stop
```

On a SLES 12 client remove the following packages if installed. This should be tried first (in case the [package]``osad`` package is not installed, do not list it on the command line):

+

```
zypper rm spacewalksd spacewalk-check zypp-plugin-spacewalk \ spacewalk-client-tools osad python2-
zypp-plugin-spacewalk \ python2-spacewalk-check python2-spacewalk-client-setup
```

- . On a SLES 15 client remove the following packages if installed:  
+

```
zypper rm spacewalk-client-setup mgr-daemon spacewalk-check \ zypp-plugin-spacewalk mgr-osad
python3-zypp-plugin-spacewalk \ python3-spacewalk-check python3-spacewalk-client-setup
```

- . You will see the following output:  
+

Refreshing service 'spacewalk'. Loading repository data... Reading installed packages... Resolving package dependencies...

The following packages are going to be REMOVED: spacewalk-check spacewalk-client-setup  
spacewalksd zypp plugin-python osad

5 packages to remove. After the operation, 301.0 KiB will be freed. Continue? [y/n/?] (y):

+

The above RPM packages are client specific, and should be removed. If this fails, they should be manually removed. The [command]``rpm -e`` command should not be used unless the [command]``zypper rm`` command above failed.

. After this is complete, the /etc/sysconfig/rhn/systemid file should be removed. This file only exists on a client machine and is used to register itself with SUSE Manager:

+

rm /etc/sysconfig/rhn/systemid

- . Any configured spacewalk channels should be deleted with:  
+

rm /etc/zypp/repos.d/spacewalk\*

- . Finally verify that repositories are properly configured. Refresh them on the server and then list them:  
+

```
zypper ref -s zypper lr
```

If any repositories pointing to spacewalk still exist, remove them with:

```
zypper repos -d zypper removerrepo <ID of the repo in the output from previous command>
```

:leveloffset: 3  
 :leveloffset: +1

**[[client.operations]]**

= 클라이언트 작업

클라이언트의 등록, 업그레이드 또는 삭제 작업뿐만 아니라 다른 작업을 수행할 수 있습니다.

{productname} 클라이언트를 개별적으로 관리하거나 시스템 세트 관리자, 시스템 그룹 또는 구성 관리를 사용하여 그룹으로 구성할 수 있습니다.

사용자 지정 시스템 정보를 얻고 구성 스냅샷을 관리하거나 SUSE Manager 웹 UI를 사용하여 클라이언트의 전원을 켜고 끄고 재부팅할 수 있습니다.

이 섹션에는 이러한 각 작업에 대한 자세한 설명이 포함되어 있습니다.

:leveloffset: 3  
 :leveloffset: +2

**[[package-management]]**

= 패키지 관리

클라이언트는 패키지를 사용해 소프트웨어를 설치, 제거 및 업그레이드합니다.

클라이언트에서 패키지를 관리하려면 [guimenu] `시스템` 으로 이동하여 관리할 클라이언트를 클릭한 후 menu:시스템[소프트웨어 > 패키지] 하위 탭으로 이동하십시오. 이 섹션에서 사용 가능한 옵션은 선택한 클라이언트의 유형과 현재 채널 구독에 따라 달라집니다.

**[IMPORTANT]**

=====

패키지가 설치되거나 업그레이드되면 라이선스 또는 EULA가 자동으로 수락됩니다.

=====

대부분의 패키지 관리 작업을 작업 체인에 추가할 수 있습니다. 작업 체인에 대한 자세한 내용은 [xref:reference:schedule/action-chains.adoc](#) [ ]을 참조하십시오.

== 패키지 검증

클라이언트에 설치한 패키지가 설치가 시작된 곳인 데이터베이스의 현재 상태와 일치해야 합니다. 설치한 패키지의 메타데이터는 파일 체크섬, 파일 크기, 권한, 소유자, 그룹, 유형 등 데이터베이스 내 정보와 비교됩니다.

#### . 절차: 설치한 패키지 검증

- . {productname} {webui}에서 [guimenu] ``시스템``으로 이동하여 패키지가 설치된 클라이언트를 클릭하고 menu:시스템[소프트웨어 > 패키지 > 검증] 하위 탭으로 이동합니다.
- . 검증하려는 패키지를 선택하고 btn:[선택한 패키지 검증]을 클릭합니다.
- . 검증이 완료되면 menu:시스템[이벤트 > 이력]으로 이동하여 결과를 확인합니다.

#### == 패키지 비교

클라이언트에 설치한 패키지를 저장한 프로파일 또는 다른 클라이언트에 설치한 패키지와 비교할 수 있습니다. 비교할 때 선택한 클라이언트가 일치하도록 수정하는 쪽을 선택할 수 있습니다.

패키지를 프로파일과 비교하려면 프로파일을 이미 저장했어야 합니다. 프로파일은 현재 설치되어 있는 클라이언트의 패키지에서 생성합니다. 프로파일을 생성했으면 이 프로파일을 사용해 동일한 패키지가 설치된 클라이언트를 추가 설치할 수 있습니다.

#### . 절차: 저장 프로파일 생성

- . {productname} {webui}에서 [guimenu] ``시스템``으로 이동하여 프로파일이 기반을 두고 있는 클라이언트를 클릭하고 menu:시스템[소프트웨어 > 패키지 > 프로파일] 하위 탭으로 이동합니다.
- . btn:[시스템 프로파일 생성]을 클릭합니다.
- . 프로파일 이름 및 설명을 입력하고 btn:[프로파일 생성]을 클릭합니다.

#### . 절차: 클라이언트 패키지 비교

- . {productname} {webui}에서 [guimenu] ``시스템``으로 이동하여 비교할 클라이언트를 클릭하고 menu:시스템[소프트웨어 > 패키지 > 프로파일] 하위 탭으로 이동합니다.
- . 저장된 프로파일과 비교하려면 프로파일을 선택하고 btn:[비교]를 클릭하십시오.
- . 다른 클라이언트와 비교하려면 클라이언트 이름을 선택하고 btn:[비교]를 클릭하여 두 클라이언트 간의 차이점 목록을 살펴봅니다.
- . 선택한 클라이언트에 설치하려는 패키지의 확인란을 선택하고, 제거하려는 패키지를 선택 취소한 다음, btn:[Sync Packages to]를 클릭합니다.

:leveloffset: 3

:leveloffset: +2

[[patch-management]]

= 패치 관리

조직 내 사용자 정의 패치를 사용해 클라이언트를 관리할 수 있습니다. 이를 통해 사용자 정의

채널에 있는 패키지에 대해 패치 경고를 발급하고, 패치 설치 일정을 잡고, 여러 조직에 걸쳐 패치를 관리할 수 있습니다.

### == 패치 생성

사용자 정의 패치를 사용하려면 패치를 생성하여 패키지를 추가하고 패치를 하나 이상의 채널에 추가하십시오.

#### . 절차: 사용자 정의 패치 생성

. {productname} {webui}에서 menu:패치[패치 관리]로 이동하여 btn:[패치 생성]을 클릭합니다.

. ``패치 생성`` 섹션에서 다음과 같은 상세 정보를 사용합니다.

+

\* ``기본 설명`` 필드에 패치에 대한 짧은 설명을 입력합니다.

\* ``권고`` 필드에 패치의 레이블을 입력합니다.

조직이 패치 관리를 더 쉽게 할 수 있도록 명명 규칙을 고안하는 것이 좋습니다.

\* ``권고 릴리스`` 필드에 패치의 릴리스 번호를 입력합니다.

예를 들어 해당 패치의 첫 버전인 경우 ``1``을 사용하십시오.

\* ``권고 유형`` 필드에서 사용할 패치의 유형을 선택합니다.

예를 들어 오류를 수정하는 패치의 경우 ``버그 수정 권고``입니다.

\* ``보안 권고``라는 권고 유형을 선택한 경우 ``권고 심각도`` 필드에서, 사용할 심각도 수준을 선택하십시오.

\* ``제품`` 필드에서 이 패치가 참조하는 제품의 이름을 입력합니다.

\* 옵션: ``원저자`` 필드에서 패치 원저자의 이름을 입력합니다.

\* ``주제``, ``설명``, ``솔루션`` 필드를 패치에 대한 추가 정보로 완성합니다.

. 옵션: ``버그`` 섹션에서 다음 상세 정보를 사용해 관련이 있는 모든 버그에 관한 정보를 지정합니다.

+

\* ``ID`` 필드에 버그 번호를 입력합니다.

\* ``요약`` 필드에 버그에 대한 짧은 설명을 입력합니다.

\* ``버그질라 URL`` 필드에 버그에 대한 짧은 설명을 입력합니다.

\* ``요약`` 필드에 버그와 관련이 있는 모든 키워드를 입력합니다.

각 키워드 사이에는 쉼표를 사용하십시오.

\* ``참조`` 및 ``메모`` 필드를 버그에 대한 추가 정보로 완성합니다.

\* 새 패치를 추가할 채널을 한 개 이상 선택합니다.

. btn:[패치 생성]을 클릭합니다.

기존 패치를 복제하여 패치를 생성할 수도 있습니다. 복제를 통해 패키지 연결이 유지되고 패치 발급이 간소화됩니다.

. 절차: 패치 복제

- . {productname} {webui}에서 menu:패치[패치 복제]로 이동합니다.
- . ``적용 가능한 패치 보기`` 필드에서 복제하려는 패치에 대한 소프트웨어 채널을 선택합니다.
- . 복제하려는 패치를 한 개 또는 여러 개 선택하고 btn:[패치 복제]를 클릭합니다.
- . 복제한 패치를 추가할 채널을 한 개 이상 선택합니다.
- . 상세 정보를 확인하여 복제를 시작합니다.

패치를 생성하였으면 이 패치에 패키지를 할당할 수 있습니다.

. 절차: 패키지를 패치에 할당

- . {productname} {webui}에서 menu:패치[패치 관리]로 이동하여 패치 상세 정보를 확인할 패치의 권고 이름을 클릭합니다.
- . menu:패키지[추가] 탭으로 이동합니다.
- . ``채널`` 필드에서 패치에 할당하려는 패키지가 포함된 소프트웨어 채널을 선택하고 btn:[패키지 보기]를 클릭합니다.
- . ``관리되는 모든 패키지`` 를 선택하여 모든 채널에서 사용 가능한 패키지를 확인할 수 있습니다.
- . 포함하려는 패키지의 확인란을 선택하고 btn:[패키지 추가]를 클릭합니다.
- . 패키지의 상세 정보를 확인하고, btn:[확인]을 클릭하여 패키지를 패치에 할당합니다.
- . menu:패키지[나열/제거] 탭으로 이동하여 패키지가 올바르게 할당되었는지 확인합니다.

패키지가 패치에 할당되면 패치 캐시가 업데이트되어 변경 사항을 반영합니다. 캐시 업데이트에 몇 분 정도 걸릴 수 있습니다.

기존 패치의 상세 정보를 변경해야 하는 경우 [guimenu] ``패치 관리`` 페이지에서 변경할 수 있습니다.

. 절차: 기존 패치 경고 편집 및 삭제

- . {productname} {webui}에서 menu:패치[패치 관리]로 이동합니다.
- . 패치의 권고 이름을 클릭하여 패치 상세 정보를 확인합니다.
- . 필요에 따라 변경하고 btn:[패치 업데이트]를 클릭합니다.
- . 패치를 삭제하려면 [guimenu] ``패치 관리`` 페이지에서 패치를 선택하고 btn:[패치 삭제]를 클릭합니다.

패치를 삭제하는 데 몇 분이 걸릴 수 있습니다.

**== 패치를 클라이언트에 적용**

패치가 준비되면 이 패치만 클라이언트에 적용하거나 다른 패치와 함께 적용할 수 있습니다.

패치 내 각 패키지는 한 개 이상의 채널에 속해 있습니다. 클라이언트가 해당 채널에 가입되어 있지 않으면 업데이트가 설치되지 않습니다.

이미 설치된 패키지보다 더 최근에 나온 버전이 클라이언트에 있는 경우 업데이트가 설치되지 않습니다. 설치된 패키지 이전에 나온 버전이 클라이언트에 있는 경우 패키지가 업그레이드됩니다.

. 절차: 해당하는 모든 패치 적용

- . {productname} {webui}에서 menu:시스템[개요]로 이동하여 업데이트하려는 클라이언트를 선택합니다.
- . menu:소프트웨어[패치] 탭으로 이동합니다.
- . btn:[모두 선택]을 클릭하여 해당하는 모든 패치를 선택합니다.
- . btn:[패치 적용]을 클릭하여 클라이언트를 업데이트합니다.

관리자 권한으로 로그인한 경우 클라이언트에 대해 더 큰 규모의 배치 업그레이드를 수행할 수도 있습니다.

. 절차: 하나의 패치를 여러 클라이언트에 적용

- . {productname} {webui}에서 menu:패치[패치 목록]으로 이동합니다.
- . 적용하려는 패치를 찾은 다음, 이 패치에 대한 ``시스템`` 열 아래의 숫자를 클릭합니다.
- . 패치를 적용하려는 클라이언트를 선택하고 btn:[패치 적용]을 클릭합니다.
- . 업데이트를 수행할 클라이언트의 목록을 확인합니다.

. 절차: 여러 패치를 여러 클라이언트에 적용

- . {productname} {webui}에서 menu:시스템[개요]로 이동하여 업데이트하려는 클라이언트를 확인하고 이 클라이언트를 시스템 세트 관리자에 추가합니다.
- . menu:시스템[시스템 세트 관리자]로 이동한 후 [guimenu]``패치`` 탭으로 이동합니다.
- . 클라이언트에 적용하려는 패치를 선택하고 btn:[패치 적용]을 클릭합니다.
- . 업데이트 실행 날짜 및 시간을 예약하고 btn:[확인]을 클릭합니다.
- . 업데이트의 진도를 확인하려면 menu:일정[보류 중인 작업]으로 이동합니다.

#### [ IMPORTANT ]

=====

예약된 패키지 업데이트는 각 클라이언트에 구성한 연결 방법을 사용해 설치됩니다. 자세한 내용은 [xref:client-configuration:contact-methods-intro.adoc](#)[ ]를 참조하십시오.

=====

:leveloffset: 3

:leveloffset: +2

[[system-locking]]

### = 시스템 잠금

시스템 잠금은 클라이언트에서 작업이 이루어지지 못하게 하는 데 사용됩니다. 예를 들어 시스템 잠금은 클라이언트가 업데이트되거나 다시 시작되지 못하게 합니다. 이 기능은 프로덕션 소프트웨어를 실행하는 클라이언트나 실수로 인한 변경 방지에 유용합니다. 작업을 수행할 준비가 되었으면 시스템 잠금을 비활성화할 수 있습니다.

시스템 잠금은 기존 클라이언트와 Salt 클라이언트에서 서로 다른 방식으로 구현됩니다.

#### == 기존 클라이언트의 시스템 잠금

기존 클라이언트가 잠겨 있으면 {webui}를 사용해 작업 일정을 잡을 수 없고, menu:시스템 [시스템 목록]에서 클라이언트의 이름 옆에 자물쇠 아이콘이 표시됩니다.

#### . 절차: 기존 클라이언트의 시스템 잠금

- . {productname} {webui}에서 잠그려는 클라이언트의 [guimenu] ``시스템 정보`` 페이지로 이동합니다.
- . [guimenu] ``잠금 상태``에서 btn:[이 시스템 잠그기]를 클릭합니다.
- btn:[이 시스템의 잠금 해제]를 클릭할 때까지 클라이언트는 잠긴 상태를 유지합니다.

```
//Something about requiring admin rights here maybe? --LKB 20200514
```

원격 명령, 자동 패치 업데이트와 같은 몇 가지 작업은 잠긴 기존 클라이언트에서 여전히 완료할 수 있습니다. 자동 패치 업데이트를 중지하려면 클라이언트의 [guimenu] ``시스템 정보`` 페이지로 이동하여 [guimenu] ``등록 정보`` 탭에서 [guimenu] ``업데이트 자동 패치``의 확인란을 선택 취소하십시오.

#### == Salt 클라이언트의 시스템 잠금

Salt 클라이언트가 잠겨 있거나 차단 모드 상태인 경우 작업의 일정을 잡을 수 없고, Salt 실행 명령이 비활성화되며, [guimenu] ``시스템 정보`` 페이지에 노란색 배너가 표시됩니다. 이 모드에서는 {webui} 또는 API를 사용해 잠긴 클라이언트에 대해 작업 일정을 잡을 수는 있지만 작업이 실패합니다.

#### [NOTE]

=====

Salt SSH 클라이언트에 대해서는 잠금 메커니즘을 사용할 수 없습니다.

=====

#### . 절차: Salt 클라이언트의 시스템 잠금

- . {productname} {webui}에서 잠그려는 클라이언트의 [guimenu] ``시스템 정보`` 페이지로 이동합니다.
- . [guimenu] ``수식`` 탭으로 이동하여 시스템 잠금 수식의 확인란을 선택한 후 btn:[저장]을 클릭합니다.
- . menu:수식[시스템 잠금] 탭으로 이동하여 [guimenu] ``시스템 잠금``의 확인란을 선택한 후 btn:[저장]을 클릭합니다.  
이 페이지에서는 클라이언트가 잠겨 있는 동안 특정 Salt 모듈을 활성화할 수도 있습니다.
- . 변경한 후 highstate를 적용해야 할 수 있습니다.  
이 경우 {webui}의 배너가 이를 알려줍니다. 시스템 잠금 수식을 제거할 때까지 클라이언트는 잠긴 상태를 유지합니다.

Salt의 차단 모드에 대한 자세한 내용은

<https://docs.saltstack.com/en/latest/topics/blackout/index.html>[ ]을 참조하십시오 .

#### == 패키지 잠금

패키지 잠금은 여러 클라이언트에서 사용할 수 있지만, 다양한 기능 집합을 사용할 수 있습니다. 구분해야 할 사항 :

- . {sle} 및 {opensuse} (zypp-based) vs {rhel} 또는 {debian} 클라이언트 및
- . 기존 클라이언트와 Salt 클라이언트 비교

#### ==== Zypp 기반 시스템의 패키지 잠금

패키지 잠금은 무단 설치나 소프트웨어 패키지로 업그레이드하는 것을 방지하는 데 사용됩니다. 패키지가 잠겨 있으면 자물쇠 아이콘이 표시되어 설치할 수 없음을 나타냅니다. 잠긴 패키지를 설치하려는 모든 시도는 이벤트 로그에 오류로 보고됩니다.

잠긴 패키지는 {productname} {webui}를 통해 또는 패키지 관리자를 사용하는 클라이언트 시스템에서 직접 설치, 업그레이드 또는 제거할 수 없습니다. 잠긴 패키지는 모든 종속 패키지도 간접적으로 잠깁니다.

#### [ NOTE ]

=====

Zypper 패키지 관리자가 있는 시스템에는 기존 및 Salt 클라이언트에서 사용할 수 있는 패키지 잠금이 있습니다.

=====

#### . 절차: 패키지 잠금 사용하기

- . 관리되는 시스템에서 menu:소프트웨어[패키지 > 잠금] 탭으로 이동하여 사용 가능한 모든 패키지의 목록을 확인합니다.
- . 잠글 패키지를 선택하고 btn:[잠금 요청]을 클릭합니다. 잠금을 활성화할 날짜와 시간을

선택합니다. 기본적으로 잠금은 가능한 한 빨리 활성화됩니다. 잠금이 즉시 활성화되지 않을 수 있습니다. 잠금이 즉시 활성화되지 않을 수 있다는 점에 유의하십시오.

- . 패키지 잠금을 제거하려면 잠금을 해제할 패키지를 선택하고 **btn:[잠금 해제 요청]**을 클릭하십시오. 잠금을 활성화할 때와 같이 날짜와 시간을 선택합니다.

==== {rhel} 및 {debian} 유사 시스템의 패키지 잠금

[ NOTE ]

=====

일부 {rhel} 및 {debian} 유사 시스템에는 Salt 클라이언트에서 사용할 수 있는 패키지 잠금이 있습니다.

=====

{rhel} 및 {debian}과 같은 시스템에서 패키지 잠금은 소프트웨어 패키지에 대한 무단 업그레이드 또는 제거하는 것을 방지하는 데만 사용됩니다. 패키지가 잠겨 있으면 자물쇠 아이콘이 표시되어 변경할 수 없음을 나타냅니다. 잠긴 패키지를 설치하려는 모든 시도는 이벤트 로그에 오류로 보고됩니다.

잠긴 패키지는 {productname} {webui}를 통해 또는 패키지 관리자를 사용하는 클라이언트 시스템에서 직접 업그레이드 또는 제거할 수 없습니다. 잠긴 패키지는 모든 종속 패키지도 간접적으로 잠깁니다.

. 절차: 패키지 잠금 사용하기

- . {rhel} 7 시스템에서 [systemitem]``root``로 [package]``yum-plugin-versionlock`` 패키지를 설치합니다. {rhel} 8 시스템에서 [systemitem]``root``로 [package]``python3-dnf-plugin-versionlock`` 패키지를 설치합니다. {debian} 시스템에서 ``apt`` 도구에는 잠금 기능이 포함되어 있습니다.
- . 관리되는 시스템에서 menu:소프트웨어[패키지 > 잠금] 탭으로 이동하여 사용 가능한 모든 패키지의 목록을 확인합니다.
- . 잠글 패키지를 선택하고 **btn:[잠금 요청]**을 클릭합니다. 잠금을 활성화할 날짜와 시간을 선택합니다. 기본적으로 잠금은 가능한 한 빨리 활성화됩니다. 잠금이 즉시 활성화되지 않을 수 있습니다. 잠금이 즉시 활성화되지 않을 수 있다는 점에 유의하십시오.
- . 패키지 잠금을 제거하려면 잠금을 해제할 패키지를 선택하고 **btn:[잠금 해제 요청]**을 클릭하십시오. 잠금을 활성화할 때와 같이 날짜와 시간을 선택합니다.

:leveloffset: 3

:leveloffset: +2

[[configuration-management]]

= 구성 관리

각 클라이언트를 수동으로 구성하는 대신에 구성 파일 및 채널을 사용해 클라이언트에 대한 구성을 관리할 수 있습니다.

구성 파라미터는 스크립트되고 구성 파일에 저장됩니다. {productname} {webui}를 사용해 직접 구성 파일을 작성할 수 있습니다. 또는 다른 위치에 있는 파일을 업로드하거나 그러한 파일에 링크할 수 있습니다.

구성 파일은 중앙에서 관리하거나 로컬에서 관리할 수 있습니다. 중앙에서 관리하는 구성 파일은 전역 구성 채널이 제공하며 {productname} 서버에 가입한 모든 클라이언트에 적용할 수 있습니다. 로컬로 관리하는 구성 파일은 중앙에서 관리하는 구성 설정을 무효화하는 데 사용되며 구성 관리 권한이 없지만 자신이 관리하는 클라이언트를 변경해야 하는 {productname} 사용자에게 특히 유용합니다.

구성 채널은 구성 파일을 체계적으로 정리하는 데 사용됩니다. 클라이언트를 구성 채널에 가입하고 필요에 따라 구성 파일을 배포할 수 있습니다.

구성 파일은 버전이 관리되므로 구성 설정을 추가하고, 이를 클라이언트에서 테스트하고, 필요에 따라 이전 리비전으로 롤백할 수 있습니다. 구성 채널을 생성했다면 다양한 구성 파일 간에, 그리고 동일한 구성 파일의 리비전 간에 비교를 수행할 수도 있습니다.

구성 파일은 중앙에서 관리하거나 로컬에서 관리할 수 있습니다. 중앙에서 관리하는 구성 파일은 전역 구성 채널이 제공합니다. 로컬에서 관리하는 구성 파일은 생성되거나 {productname}로 직접 업로드됩니다.

Salt 클라이언트와 기존 클라이언트에 사용할 수 있는 구성 관리 기능은 서로 다릅니다. 아래 표에는 서로 다른 클라이언트 유형에서 지원되는 기능이 표시되어 있습니다.

#### .지원되는 구성 관리 기능

```
[cols="1,1,1", options="header"]
```

```
| ===
```

```
| 기능
```

```
| Salt
```

```
| 기존
```

```
| 전역 구성 채널
```

```
| {check}
```

```
| {check}
```

```
| 파일 배포
```

```
| {check}
```

```
| {check}
```

```
| 파일 비교
```

```
| {question}
```

```
| {check}
```

```
| 로컬에서 관리하는 파일
```

```
| {cross}
```

```

| {check}
| 샌드박스 파일
| {cross}
| {check}

| Highstate 적용
| {check}
| {cross}

| 클라이언트에서 파일 임포트
| {cross}
| {check}

| 구성 매크로
| {cross}
| {check}

```

| ===

// Edited these for style, not tested. --LKB 2020-07-31

== 구성 관리를 위한 기존 클라이언트 준비

구성 관리를 사용하려면 기존 클라이언트에 몇 가지 추가 준비 작업이 필요합니다. {ay} 또는 {kickstart}로 기존 클라이언트를 설치했다면 아마도 적절한 패키지가 이미 있을 것입니다. 다른 기존 클라이언트의 경우 클라이언트 운영 체제에 대해 관련 도구 하위 채널을 설치했는지 확인하십시오. 소프트웨어 채널에 대한 자세한 내용은 [xref:client-configuration:channels.adoc\[ \]](#)를 참조하십시오.

필요한 패키지는 다음과 같습니다.

- \* [path]``mgr-cfg``: 모든 [path]``mgr-cfg-\*`` 패키지에 필요한 기본 라이브러리 및 함수
- \* [path]``mgr-cfg-actions``: {productname}를 사용하여 예약된 구성 작업을 실행하기 위해 필요합니다.
- \* [path]``mgr-cfg-client``: 구성 관리 시스템의 클라이언트 기능에 명령줄 인터페이스 제공
- \* [path]``mgr-cfg-management``: {productname} 구성을 관리할 수 있는 명령줄 인터페이스 제공

menu:시스템[활성화 키]로 이동하여 부트스트랩 중에 사용하려는 활성화 키를 클릭하고 [guimenu]``구성 파일 배포`` 옵션 확인란을 선택하여 부트스트랩 프로세스 중에 이 패키지를 설치할 수 있습니다. 활성화 키에 대한 자세한 내용은 [xref:client-configuration:activation-keys.adoc\[ \]](#)를 참조하십시오.

## == 구성 채널 생성

새로운 중앙 구성 채널을 생성하려면 다음과 같이 하십시오.

### . 절차: 중앙 구성 채널 생성

- . {productname} {webui}에서 menu:구성[채널]로 이동하여 btn:[구성 채널 생성]을 클릭합니다.
  - . 채널의 이름을 입력합니다.
  - . 채널의 레이블을 입력합니다.
- 이 필드는 글자, 숫자, 하이픈(``-``), 밑줄(``\_``)만 포함해야 합니다.
- . 다른 채널과 구별할 수 있게 해주는 채널 설명을 입력합니다.
  - . btn:[구성 채널 생성]을 클릭하여 새 채널을 생성합니다.

구성 채널을 사용해 Salt 클라이언트에서 Salt 상태를 관리할 수도 있습니다.

### . 절차: Salt 상태 채널 생성

- . {productname} {webui}에서 menu:구성[채널]로 이동하여 btn:[상태 채널 생성]을 클릭합니다.
  - . 채널의 이름을 입력합니다.
  - . 채널의 레이블을 입력합니다.
- 이 필드는 글자, 숫자, 하이픈(``-``), 밑줄(``\_``)만 포함해야 합니다.
- . 다른 채널과 구별할 수 있게 해주는 채널 설명을 입력합니다.
  - . [path]``init.sls`` 파일에 대해 [guimenu]``SLS 컨텐트``를 입력합니다.
  - . btn:[구성 채널 생성]을 클릭하여 새 채널을 생성합니다.

## == 구성 파일, 디렉토리 또는 심볼 링크 추가

구성 채널을 생성했으면 이제 구성 파일, 디렉토리 또는 심볼 링크를 추가할 수 있습니다.

### . 절차: 구성 파일, 디렉토리 또는 심볼 링크 추가

- . {productname} {webui}에서 menu:구성[채널]로 이동하여 구성 파일을 추가하려는 구성 채널의 이름을 클릭한 후, menu:파일 추가[파일 생성] 하위 탭으로 이동합니다.
- . [guimenu]``파일 형식`` 필드에서 텍스트 파일, 디렉토리, 심볼 링크 중 생성하려는 것을 선택합니다.
- . [path]``파일 이름/경로`` 필드에 파일을 배포해야 하는 위치의 절대 경로를 입력합니다.
- . 심볼 링크를 생성하려면 [guimenu]``심볼 링크 대상 파일 이름/경로`` 필드에 대상 파일 및 경로를 입력하십시오.
- . [guimenu]``소유권`` 필드에서 파일의 [guimenu]``사용자 이름`` 및 [guimenu]``그룹 이름``을 입력하고 [guimenu]``파일 권한 모드``를 입력합니다.
- . 클라이언트가 SELinux를 활성화한 경우 [guimenu]``SELinux 컨텍스트``를 구성하여 필수

파일 속성(예: 사용자, 역할, 파일 형식)을 활성화할 수 있습니다.

- . 구성 파일에 매크로가 포함된 경우 매크로의 시작과 끝을 표시하는 기호를 입력하십시오.
- . [guimenu]``파일 내용`` 텍스트 상자에 구성 파일 내용을 입력합니다. 이때 스크립트 드롭다운 상자를 사용해 적절한 스크립팅 언어를 선택합니다.
- . btn:[구성 파일 생성]을 클릭합니다.

== 클라이언트를 구성 채널에 가입

menu:시스템[시스템 목록]으로 이동하여 가입하려는 클라이언트를 선택하고 [guimenu]``구성``탭으로 이동하여 개별 클라이언트를 구성 채널에 가입할 수 있습니다. 여러 클라이언트를 구성 채널에 가입하려면 시스템 세트 관리자(SSM)를 사용하면 됩니다.

. 절차: 여러 클라이언트를 구성 채널에 가입

- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 작업하려는 클라이언트를 선택합니다.
- . menu:시스템[시스템 세트 관리자]로 이동하여 menu:구성[채널에 가입] 하위 탭에서 사용 가능한 구성 채널의 목록을 확인합니다.
- . 옵션: [guimenu]``현재 가입된 시스템`` 열의 숫자를 클릭하여 현재 어떤 클라이언트가 구성 채널에 가입되어 있는지 확인합니다.
- . 가입하려는 구성 채널을 확인하고 btn:[계속]을 클릭합니다.
- . 위 및 아래 방향 화살표를 사용해 구성 채널의 순위를 지정합니다.  
설정이 구성 채널 간에 서로 충돌하는 경우 목록 상단에 더 가까이 있는 채널이 우선합니다.
- . 선택한 클라이언트에 채널이 적용되는 방식을 결정합니다.  
btn:[최하위 우선순위로 가입]을 클릭하여 새 채널을 현재 가입된 채널보다 낮은 우선순위로 추가합니다. btn:[최상위 우선순위로 가입]을 클릭하여 새 채널을 현재 가입된 채널보다 높은 우선순위로 추가합니다. btn:[기존 구독 대체]를 클릭하여 기존 채널을 제거하고 새 채널로 교체합니다.
- . btn:[구독 적용]을 클릭합니다.

#### [NOTE]

=====

새 구성 채널 우선순위가 기존 채널과 충돌하는 경우 중복 채널이 제거되고 새로운 우선순위에 따라 교체됩니다. 작업에 의해 클라이언트의 구성 우선순위가 변경되는 경우 {webui}는 작업을 진행하기 전에 이러한 변경을 확인하도록 사용자에게 요구합니다.

=====

== 구성 파일 비교

또한 시스템 세트 관리자(SSM)를 사용해 클라이언트에 배포된 구성 파일을 {productname} 서버에 저장된 구성 파일과 비교할 수 있습니다.

#### . 절차: 구성 파일 비교

- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 비교하려는 구성 파일에 가입된 클라이언트를 선택합니다.
- . menu:시스템[시스템 세트 관리자]로 이동하여 menu:구성[파일 비교] 하위 탭에서 사용 가능한 구성 채널의 목록으로 이동합니다.
- . 옵션: [guimenu]``시스템`` 열의 숫자를 클릭하여 현재 어떤 클라이언트가 구성 파일에 가입되어 있는지 확인합니다.
- . 비교할 구성 파일을 확인하고 btn:[파일 비교 작업 일정 잡기]를 클릭합니다.

#### == 기존 클라이언트의 구성 파일 매크로

1개의 파일을 저장하고 동일한 구성을 공유할 수 있다는 점이 유용하지만, 경우에 따라 동일한 구성 파일의 여러 변형이 필요하거나 호스트 이름 및 MAC 주소와 같은 시스템별 세부 정보만 다른 구성 파일이 필요할 수 있습니다. 이러한 경우에는 구성 파일 내에서 매크로 또는 변수를 사용할 수 있습니다. 이 작업을 통해 변형이 수백 또는 수천 개인 1개의 파일을 업로드 및 배포할 수 있습니다. 사용자 정의 시스템 정보를 위한 변수뿐만 아니라 다음 표준 매크로도 지원됩니다.

|                                                       |                         |                                              |                                  |
|-------------------------------------------------------|-------------------------|----------------------------------------------|----------------------------------|
| rhn.system.sid                                        | rhn.system.profile_name | rhn.system.description                       | rhn.system.hostname              |
| rhn.system.ip_address                                 |                         |                                              | rhn.system.custom_info(key_name) |
| rhn.system.net_interface.ip_address(eth_device)       |                         | rhn.system.net_interface.netmask(eth_device) |                                  |
| rhn.system.net_interface.broadcast(eth_device)        |                         |                                              |                                  |
| rhn.system.net_interface.hardware_address(eth_device) |                         |                                              |                                  |
| rhn.system.net_interface.driver_module(eth_device)    |                         |                                              |                                  |

이 기능을 사용하려면 [guimenu]``구성 채널 정보`` 페이지를 통해 구성 파일을 업로드하거나 생성합니다. 그리고 [guimenu]``구성 파일 정보`` 페이지를 열고 선택한 지원 매크로를 포함합니다. 변수를 오피셋하기 위해 사용되는 구분 기호가 [guimenu]``매크로 시작 구분 기호`` 및 [guimenu]``매크로 끝 구분 기호`` 필드에 설정된 구분 기호와 일치하고 파일의 다른 문자와 충돌하지 않는지 확인합니다. 구분 기호의 길이는 2자이며 퍼센트(``%``) 기호를 포함하지 않는 것이 좋습니다.

예를 들어, IP 주소와 호스트 이름만 다른 모든 서버에 적용할 수 있는 파일이 있을 수 있습니다. 각 서버에 대해 별도의 구성 파일을 관리하는 대신 IP 주소 및 호스트 이름 매크로가 포함된 1개의 파일(예: [path]``server.conf``)을 생성할 수 있습니다.

```
hostname={| rhn.system.hostname |} ip_address={| rhn.system.net_interface.ip_address(eth0) |}
```

파일이 {productname} {webui} 또는 {productname} 구성 클라이언트([command]``mgrcfg-client``)의 명령줄에서 예약된 작업을 통해 개별 시스템에 전달되는 경우 변수는 {productname}의 시스템 프로파일에 기록된 시스템의 호스트 이름 및 IP

주소로 대체됩니다. 이 예에서 배포된 버전은 다음과 같습니다.

```
hostname=test.example.domain.com ip_address=177.18.54.7
```

사용자 정의 시스템 정보를 수집하려면 사용자 정의 정보

매크로(` `rhn.system.custom\_info``)에 키 레이블을 삽입합니다. 예를 들어, "asset"이라는 레이블의 키를 개발한 경우 구성 파일의 사용자 정의 정보 매크로에 추가하여 이 키를 포함한 모든 시스템에서 값을 대체할 수 있습니다. 매크로는 다음과 같습니다.

```
asset=@ rhn.system.custom_info(asset) @}
```

해당 키에 대한 값을 포함한 시스템으로 파일이 배포되면 매크로가 번역되어 다음과 유사한 문자열이 제공됩니다.

```
asset=Example#456
```

예를 들어, 오류를 방지하기 위해 기본값이 필요한 경우 기본값을 포함하려면 다음과 같이 사용자 정의 정보 매크로에 추가할 수 있습니다.

```
asset=@ rhn.system.custom_info(asset) = 'Asset #' @}
```

이 기본값은 해당 값을 포함한 모든 시스템의 값으로 덮어쓰기됩니다.

{productname} 구성 관리자([command] ``mgrcfg-manager``)는 {productname} 클라이언트 시스템에서 시스템 관리를 지원하기 위해 사용할 수 있습니다. 이 도구는 시스템 기반이 아니므로 파일을 변환하거나 수정하지 않습니다. [command] ``mgrcfg-manager`` 명령은 시스템 설정과 관계가 없습니다. 바이너리 파일은 보간할 수 없습니다.

```
:leveloffset: 3
:leveloffset: +2
```

[ [power-management]]  
= 전원 관리

{productname} {webui}를 사용해 클라이언트를 켜고, 끄고, 다시 부팅할 수 있습니다.

이 기능은 IPMI 또는 Redfish 프로토콜을 사용하며 Cobbler 프로파일을 사용해 관리됩니다. 선택한 클라이언트에는 이러한 프로토콜 중 하나를 지원하는 전원 관리 컨트롤러가 있어야 합니다.

Redfish의 경우 클라이언트와 {productname} 서버 간의 타당한 SSL 연결을 설정할 수 있는지 확인하십시오. 사용자는 Redfish 관리 컨트롤러의 SSL 서버 인증서에 서명하는 데 사용되는 인증 주체를 신뢰했어야 합니다. CA 인증서는 ``.pem`` 형식이어야 하며

{productname} 서버의 [path]``/etc/pki/trust/anchors/``에 저장됩니다. 인증서 저장을 완료했으면 [command]``update-ca-certificate``을 실행하십시오.

- . 절차: 전원 관리 활성화
- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 관리하려는 클라이언트를 선택하고 menu:조달[전원 관리] 탭으로 이동합니다.
- . [guimenu]``유형`` 필드에서 사용할 전원 관리 프로토콜을 선택합니다.
- . 전원 관리 서버의 상세 정보를 모두 입력하고, 취할 조치에 해당하는 적절한 버튼을 클릭하거나 btn:[저장만]을 클릭하여 아무 조치도 취하지 않고 상세 정보를 저장합니다.

여러 클라이언트를 시스템 세트 관리자에 추가함으로써 전원 관리 조치를 여러 클라이언트에 동시에 적용할 수 있습니다. 시스템 세트 관리자에 대한 자세한 내용은 [xref:client-configuration:system-set-manager.adoc](#)[ ]를 참조하십시오.

## == 전원 관리 및 Cobbler

전원 관리 기능을 처음 사용할 때 Cobbler 시스템 레코드가 자동으로 생성됩니다(클라이언트에 아직 없는 경우). 이렇게 자동으로 생성된 시스템 레코드는 네트워크에서 부팅할 수 있고 더미 시스템 이미지에 대한 참조를 포함합니다. 이 참조가 필요한 이유는 현재 Cobbler가 프로파일 또는 이미지 없는 시스템 레코드를 지원하지 않기 때문입니다.

Cobbler 전원 관리는 펜스 에이전트 도구를 사용해 IPMI가 아닌 프로토콜을 지원합니다. {productname}은 IPMI 및 Redfish 프로토콜만 지원합니다. [path]``rhn.conf`` 구성 파일의 [option]``java.power\_management.types`` 구성 파라미터에 펜스 에이전트 이름을 쉼표로 구분된 목록으로 추가하여 클라이언트가 다른 프로토콜을 사용하도록 구성할 수 있습니다.

```
:leveloffset: 3
:leveloffset: +2
```

```
[[snapshots]]
= 구성 스냅샷
```

스냅샷은 설정된 시점에 클라이언트에 대한 패키지 프로파일, 구성 파일 및 {productname} 설정을 기록합니다. 이전 스냅샷으로 롤백하여 이전 구성 설정을 복원할 수 있습니다.

[NOTE]

=====

스냅샷은 기존 클라이언트에서만 지원합니다. Salt 클라이언트는 이 기능을 지원하지 않습니다.

=====

스냅샷은 일부 작업이 수행된 후 자동으로 캡처됩니다. 언제든 수동으로 스냅샷을 촬영할 수도 있습니다. 클라이언트에서 잠재적으로 파괴적인 작업을 수행하기 전에 현재 스냅샷을 확보하는 것이 좋습니다.

스냅샷은 기본적으로 활성화되어 있습니다. [path]``rhn.conf`` 구성 파일에서 [parameter]``enable\_snapshots=0``으로 설정하여 자동 스냅샷을 비활성화할 수 있습니다.

menu:시스템[시스템 목록]으로 이동해 관리하려는 클라이언트를 선택하여 스냅샷을 관리합니다.  
 menu:조달[스냅샷] 탭으로 이동하여 선택한 클라이언트의 모든 현재 스냅샷 목록을 확인합니다.  
 스냅샷의 이름을 클릭하여 스냅샷에 기록된 변경 사항을 더 자세히 알아봅니다. 선택한 스냅샷에 대한 롤백으로 인해 다음 항목에 발생하는 변경 사항을 menu:조달[스냅샷] 탭의 하위 탭을 사용해 확인할 수 있습니다.

- \* 그룹 멤버쉽
- \* 채널 서브스크립션
- \* 패키지 설치됨
- \* 구성 채널 구독
- \* 구성 파일
- \* 스냅샷 태그

#### [ IMPORTANT ]

=====

스냅샷을 사용해 변경 사항 전체가 아닌 대부분을 클라이언트에 롤백할 수 있습니다. 예를 들어 여러 업데이트를 롤백할 수 없고, 제품 마이그레이션을 롤백할 수 없습니다. 클라이언트에서 업그레이드를 수행하기 전에 백업을 완료했는지 항상 확인하십시오.

=====

#### == 스냅샷 태그

스냅샷 태그를 통해 스냅샷에 의미 있는 설명을 추가할 수 있습니다. 태그를 사용해 마지막으로 알려진 작업 구성 또는 성공한 업그레이드 등 스냅샷에 대한 추가 정보를 기록할 수 있습니다.

menu:시스템[시스템 목록]으로 이동해 관리하려는 클라이언트를 선택하여 스냅샷 태그를 관리합니다. menu:조달[스냅샷 태그] 탭으로 이동하여 선택한 클라이언트의 모든 현재 스냅샷 태그 목록을 확인합니다. [guimenu]``시스템 태그 생성``을 클릭하고, 설명을 입력한 후 btn:[현재 스냅샷에 태그 지정] 버튼을 클릭합니다.

#### == 대규모 설치 시 스냅샷

{productname}가 유지할 수 있는 스냅샷의 최대 수는 없습니다. 따라서 스냅샷을 저장하는 데이터베이스는 클라이언트, 패키지, 채널 및 구성 변경 사항을 추가함에 따라 규모가 커집니다.

클라이언트가 수천 개인 대규모 설치의 경우 {productname} API를 사용해 정기적 일정에 정기적인 정리 스크립트를 생성하여 이전 스냅샷이 정기적으로 삭제되게 할 수 있습니다. 또는 [path]``rhn.conf`` 구성 파일에서 [parameter]``enable\_snapshots=0``으로 설정하여 이 기능을 비활성화할 수 있습니다.

```
:leveloffset: 3
:leveloffset: +2
```

**[[custom-info]]**  
= 사용자 정의 시스템 정보

클라이언트에 대한 사용자 정의 시스템 정보를 포함할 수 있습니다. 시스템 정보는 클라이언트에 할당할 수 있는 키:값 페어로 정의됩니다. 예를 들어 특정 프로세서에 대한 키:값 페어를 정의한 다음, 이 키를 프로세서가 설치된 모든 클라이언트에 할당할 수 있습니다. 사용자 정의 시스템 정보는 분류되며 {productname} {webui}를 사용해 검색할 수 있습니다.

시작하기 전에 사용자 정의를 저장할 수 있게 허용하는 키를 생성해야 합니다.

- . 절차: 사용자 정의 시스템 정보 키 생성
  - . {productname} {webui}에서 menu:시스템[사용자 정의 시스템 정보]로 이동하여 btn:[키 생성]을 클릭합니다.
  - . [guimenu]``키 레이블`` 필드에 키의 이름을 추가합니다. 공백을 사용하면 안 됩니다. 예를 들어, ``intel-x86\_64-quadcore``를 추가합니다.
  - . [guimenu]``설명`` 필드에 필요한 추가 정보를 입력합니다.
  - . 필요한 각 키에 이 작업을 반복합니다.

For Salt clients, this information is available via Salt pillar. You can retrieve this information from a Salt client with a command such as:

```
salt $minionid pillar.get custom_info:key1
```

이 명령을 실행한 후의 출력은 다음과 같습니다.

```
$minionid: val1
```

사용자 정의 시스템 정보 키를 몇 개 생성했으면 이 키를 클라이언트에 적용할 수 있습니다.

- . 절차: 사용자 정의 정보 키를 클라이언트에 적용
  - . {productname} {webui}에서 [guimenu]``시스템``으로 이동하여 사용자 정의 정보를 적용할 클라이언트를 클릭하고, menu:상세 정보[사용자 정의 정보] 탭으로 이동합니다.
  - . btn:[값 생성]을 클릭합니다.
  - . 적용하려는 값을 찾아 키 레이블을 클릭합니다.
  - . [guimenu]``값`` 필드에 추가 정보를 입력합니다.
  - . btn:[키 업데이트]를 클릭하여 사용자 정의 정보를 클라이언트에 적용합니다.

구성 관리에 대한 자세한 설명은 [xref:client-configuration:configuration-management.adoc](#)[ ]에서 참조하십시오.

```
:leveloffset: 3
:leveloffset: +2
```

**[ [ssm] ]**

= 시스템 세트 관리자

시스템 세트 관리자(SSM)는 한 번에 두 개 이상의 클라이언트에서 작업을 수행하는 데 사용됩니다. SSM은 수명이 짧은 클라이언트 세트를 생성하므로 다수의 클라이언트에 적용해야 하는 켜기-끄기 작업에 유용합니다. 더 오래 지속되는 세트를 원하는 대신에 시스템 그룹을 사용하는 것을 고려하십시오. 시스템 그룹에 대한 자세한 내용은 [xref:client-configuration:system-groups.adoc](#)[ ]를 참조하십시오.

SSM에 사용할 수 있는 작업이 아래 표에 나열되어 있습니다. 이 표의 아이콘에는 다음과 같은 의미가 있습니다.

- \* {check} 이 작업은 이 클라이언트 유형의 SSM에서 사용할 수 있습니다.
- \* {cross} 이 작업은 이 클라이언트 유형의 SSM에서 사용할 수 없습니다.
- \* {question} 이 작업은 이 클라이언트 유형에 대해 고려 중이며, 이후에 지원될 수도 있고 지원되지 않을 수도 있습니다.

#### . 사용할 수 있는 SSM 작업

```
[cols="1,1,1", options="header"]
```

```
| ==
```

| 작업             | 기존      | Salt    |
|----------------|---------|---------|
| 시스템 나열         | {check} | {check} |
| 패치 설치          | {check} | {check} |
| 패치 업데이트 일정 잡기  | {check} | {check} |
| 패키지 업그레이드      | {check} | {check} |
| 패키지 설치         | {check} | {check} |
| 패키지 제거         | {check} | {check} |
| 패키지 검증         | {check} | {cross} |
| 그룹 생성          | {check} | {check} |
| 그룹 관리          | {check} | {check} |
| 채널 멤버쉽         | {check} | {check} |
| 채널 가입          | {check} | {cross} |
| 배포/여러 채널       | {check} | {cross} |
| 클라이언트 자동 설치    | {check} | {cross} |
| 스냅샷에 대한 태그     | {check} | {cross} |
| 원격 명령          | {check} | {cross} |
| 전원 관리          | {check} | {cross} |
| 시스템 기본 설정 업데이트 | {check} | {check} |

|                       |         |         |
|-----------------------|---------|---------|
| 하드웨어 프로파일 업데이트        | {check} | {check} |
| 패키지 프로파일 업데이트         | {check} | {check} |
| 사용자 정의 값 설정/제거        | {check} | {check} |
| 클라이언트 재부팅             | {check} | {check} |
| 클라이언트를 다른 조직으로 마이그레이션 | {check} | {check} |
| 클라이언트 삭제              | {check} | {check} |

| ===

SSM에 대한 클라이언트를 다음 몇 가지 방식으로 선택할 수 있습니다.

- \* menu:시스템[시스템 목록]으로 이동하여 작업하려는 클라이언트의 확인란을 선택합니다.
- \* menu:시스템[시스템 그룹]으로 이동하여 작업하려는 시스템 그룹에 대해 btn:[SSM에서 사용]을 클릭합니다.
- \* menu:시스템[시스템 그룹]으로 이동하여 작업하려는 그룹의 확인란을 선택하고, btn:[그룹 작업]을 클릭합니다.

작업하려는 클라이언트를 선택했으면 menu:시스템[시스템 세트 관리자]로 이동하거나 상단 메뉴 모음에서 [guimenu]``선택한 시스템`` 아이콘을 클릭합니다.

#### [ NOTE ]

=====

SSM의 상세 정보는 {productname} {webui}의 다른 부분에 있는 상세 정보와 약간 다를 수 있습니다. SSM에 사용할 수 있는 모든 업데이트가 표시됩니다. 따라서 최신 버전이 아닐 수 있는 패키지로 업그레이드할 수 있습니다.

=====

== SSM에서 기본 채널 변경하기

SSM을 사용해 동시에 클라이언트 두 개 이상의 기본 채널을 변경할 수 있습니다.

#### [ IMPORTANT ]

=====

기본 채널을 상당 부분 변경하면 영향을 받는 클라이언트에서 사용할 수 있는 패키지 및 패치가 달라집니다. 주의해서 사용하십시오.

=====

. 절차: SSM을 사용해 여러 클라이언트의 기본 채널 변경

- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 작업하려는 클라이언트의 확인란을 선택하고 menu:시스템[시스템 세트 관리자]로 이동합니다.
- . [guimenu]``채널`` 하위 탭으로 이동합니다.
- . 목록에서 현재 기본 채널을 찾고, [guimenu]``시스템`` 열에 표시된 숫자가 정확한지

확인합니다.

이 열의 숫자를 클릭하면 변경하려는 클라이언트에 관한 더 자세한 정보를 볼 수 있습니다.

- . [guimenu] ``원하는 기본 채널`` 필드에서 새 기본 채널을 선택하고, btn:[다음]을 클릭합니다.

- . 각 하위 채널에 대해 [guimenu] ``변경 사항 없음``, [guimenu] ``가입`` 또는 [guimenu] ``가입 취소``를 선택하고, btn:[다음]을 클릭합니다.

- . 변경하려는 사항을 확인하고, 작업이 수행되는 시점을 선택합니다.

- . btn:[확인]을 클릭하여 변경 사항을 예약합니다.

:leveloffset: 3

:leveloffset: +2

[[system-groups]]

= 시스템 그룹

시스템 그룹을 사용해 다수의 클라이언트를 더 쉽게 관리할 수 있습니다. 그룹을 사용해 업데이트, 구성 채널, salt 상태 또는 수식의 적용과 같은 대량 작업을 클라이언트에서 수행할 수 있습니다.

환경에 적합한 방식으로 클라이언트를 그룹으로 편성할 수 있습니다. 예를 들어 클라이언트를 설치되는 운영 체제, 있어야 할 물리적 위치 또는 처리하는 워크로드 유형에 따라 체계적으로 편성할 수 있습니다. 클라이언트가 속할 수 있는 그룹의 수에는 제한이 없으므로 그룹을 다양한 방식으로 정의할 수 있습니다.

클라이언트를 그룹에 편성했으면 이제 하나 이상의 그룹에 속한 모든 클라이언트 또는 그룹 간 교집합에서 업데이트를 수행할 수 있습니다. 예를 들어 모든 Salt 클라이언트에 대한 그룹과 모든 SLES 클라이언트에 대한 그룹을 정의할 수 있습니다. 그런 다음, 모든 Salt 클라이언트에서 업데이트를 수행하거나 그룹 간 교집합을 사용해 모든 Salt SLES 클라이언트를 업데이트할 수 있습니다.

== 그룹 생성

몇 개의 그룹을 먼저 생성해야 이 그룹을 사용해 클라이언트를 편성할 수 있습니다.

. 절차: 새 시스템 그룹 생성

- . {productname} {webui}에서 menu:시스템[시스템 그룹]으로 이동합니다.
- . btn:[그룹 생성]을 클릭합니다.
- . 새 그룹에 이름 및 설명을 부여합니다.
- . btn:[그룹 생성]을 클릭하여 그룹을 저장합니다.
- . 필요한 각 그룹에 이 작업을 반복합니다.

== 클라이언트를 그룹에 추가

개별 클라이언트를 그룹에 추가하거나 여러 클라이언트를 동시에 추가할 수 있습니다.

. 절차: 단일 클라이언트를 그룹에 추가

- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 추가할 클라이언트의 이름을 클릭합니다.
- . menu:그룹[조인] 탭으로 이동합니다.
- . 조인할 그룹의 확인란을 선택하고 btn:[선택한 그룹 조인]을 클릭합니다.

. 절차: 여러 클라이언트를 그룹에 추가

- . {productname} {webui}에서 menu:시스템[시스템 그룹]으로 이동하여 클라이언트를 추가할 그룹의 이름을 클릭합니다.
- . [guimenu]``대상 시스템`` 탭으로 이동합니다.
- . 추가할 클라이언트의 확인란을 선택하고 btn:[시스템 추가]를 클릭합니다.

. 절차: SSM으로 여러 클라이언트를 그룹에 추가

- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 추가할 각 클라이언트의 확인란을 선택하면 클라이언트가 시스템 세트 관리자에 추가됩니다.
- . menu:시스템[시스템 세트 관리자]로 이동한 후 [guimenu]``그룹`` 탭으로 이동합니다.
- . 조인할 그룹을 찾아 [guimenu]``추가``의 확인란을 선택합니다.
- . btn:[멤버쉽 변경]을 클릭합니다.
- . btn:[확인]을 클릭하여 선택한 그룹에 클라이언트를 조인합니다.

시스템 세트 관리자에 대한 자세한 내용은 [xref:client-configuration:system-set-manager.adoc](#)[ ]에서 확인할 수 있습니다.

menu:시스템[시스템 그룹]으로 이동하여 그룹의 이름을 클릭한 후 [guimenu]``시스템`` 탭으로 이동하여 어떤 클라이언트가 그룹에 있는지 확인할 수 있습니다. 또는 menu:시스템[시작화 > 시스템 그룹화]로 이동하여 시스템 그룹을 시각적으로 표현한 것을 볼 수 있습니다.

## == 그룹 작업

클라이언트를 그룹에 배치하고 나면 그룹을 사용해 업데이트를 관리할 수 있습니다. Salt 클라이언트의 경우 그룹 내 모든 클라이언트에 상태와 수식을 적용할 수도 있습니다.

{productname} {webui}에서 menu:시스템[시스템 그룹]으로 이동합니다. 그룹 내 클라이언트 중 어느 하나에 대해서라도 사용할 수 있는 업데이트가 있는 경우 목록에 아이콘이 표시됩니다. 사용 가능한 업데이트에 대한 자세한 정보를 확인하고 업데이트를 클라이언트에 적용하려면 아이콘을 클릭합니다.

한 번에 두 개 이상의 그룹으로 작업할 수도 있습니다. 작업할 그룹을 선택하고 **btn:[합집합 작업]**을 클릭하여 선택한 각 그룹에서 각 클라이언트를 선택합니다.

또는 그룹의 교집합에서 작업할 수 있습니다. 두 개 이상의 그룹을 선택하고, **btn:[교집합 작업]**을 클릭하여 선택한 모든 그룹에 있는 클라이언트만 선택하십시오. 예를 들어 모든 Salt 클라이언트에 대한 그룹, 모든 SLES 클라이언트에 대한 그룹이 하나씩 있을 수 있습니다. 이 그룹의 교집합은 모든 Salt SLES 클라이언트가 됩니다.

```
:leveloffset: 3
:leveloffset: +2
```

```
[[system-types]]
= 시스템 유형
```

클라이언트는 시스템 유형을 기준으로 분류됩니다. 각 클라이언트에는 기본 시스템 유형과 할당된 추가 시스템 유형이 있을 수 있습니다.

기본 시스템 유형은 기존 클라이언트의 경우 ``관리``, Salt 클라이언트의 경우 ``Salt``를 포함합니다.

추가 시스템 유형에는 가상 호스트로 작동하는 클라이언트의 경우 ``가상화 호스트``, 빌드 호스트로 작동하는 클라이언트의 경우 ``컨테이너 빌드 호스트``를 포함합니다.

**menu:시스템[시스템 목록 > 시스템 유형]**으로 이동하여 추가 시스템 유형을 조정할 수 있습니다. 추가 시스템 유형을 변경하려는 클라이언트의 확인란을 선택하고, **[guimenu]``추가 시스템 유형``**을 선택하고, **btn:[시스템 유형 추가]** 또는 **btn:[시스템 유형 제거]**를 클릭합니다.

클라이언트를 다시 등록하여 기본 시스템 유형을 ``관리``에서 ``Salt``로 변경할 수도 있습니다.

**[WARNING]**

=====

기본 시스템 유형을 변경하려면 클라이언트를 다시 등록해야 합니다. 다시 등록할 경우 클라이언트에서 모든 사용자 정의나 구성이 삭제되지만 이벤트 이력은 보존됩니다. 클라이언트 작동 중지 시간도 필요합니다.

=====

== {webui}를 사용해 기존 클라이언트에서 Salt로 변경

기존 클라이언트를 Salt 클라이언트로 변경하는 가장 간단한 방법은 {webui}로 다시 등록하는 것입니다.

//[WARNING]

//=====

//Changing the base system type requires that you re-register your client.

```
//This deletes any customization or configuration on the client, however
event history is preserved.
//It also requires client downtime.
//=====
```

// Not tested --LKB 2020-09-22

- . 절차: {webui}를 사용해 기존 클라이언트에서 Salt로 변경
- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 변경하려는 클라이언트를 식별하고, 호스트 이름을 적어 둡니다.
- . menu:시스템[부트스트랩]으로 이동합니다.
- . [guimenu] ``호스트`` 필드에 다시 등록할 클라이언트의 호스트 이름을 입력합니다.
- . 필요에 따라 다른 필드를 완성합니다.
- . btn:[부트스트랩]을 클릭하여 부트스트랩 프로세스의 일정을 잡습니다.

+

클라이언트는 등록을 완료한 후 [guimenu] ``시스템 목록``에 ``Salt``라는 시스템 유형과 함께 표시됩니다.

== 명령 프롬프트에서 기존 클라이언트를 Salt로 변경

명령 프롬프트를 사용해 기존 클라이언트를 Salt 클라이언트로 다시 등록할 수 있습니다. 이렇게 하려면 기존 클라이언트가 사용하는 패키지를 삭제해야 합니다. 그런 다음, Salt 클라이언트에 대해 선호하는 등록 방법을 사용해 클라이언트를 다시 등록하면 됩니다.

//[WARNING]

//=====

```
//Changing the base system type requires that you re-register your
client.
//This deletes any customization or configuration on the client.
//It also requires client downtime.
//=====
```

// Not tested --LKB 2020-09-22

- . 절차: 명령 프롬프트에서 기존 클라이언트를 Salt로 변경
- . 변경할 클라이언트의 명령 프롬프트에서 패키지 관리자를 사용해 다음 패키지를 제거합니다.

+

spacewalk-check spacewalk-client-setup osad osa-common mgr-osad spacewalksd mgr-daemon rhnlib  
rhnmd

- . 선호하는 등록 방법을 사용해 클라이언트를 Salt 클라이언트로 다시 등록합니다.

+

클라이언트는 등록을 완료한 후 [guimenu] ``시스템 목록``에 ``Salt``라는 시스템 유형과

함께 표시됩니다.

```
:leveloffset: 3
:leveloffset: +1

[[autoinstallation]]
= 운영 체제 설치
```

일반적으로 이미 실행 중인 클라이언트를 등록합니다. 이러한 시스템을 수동으로 설치한 직후 *{productname}*를 등록하거나 이미 설치된 기존 시스템의 환경에 *{productname}*를 추가할 수 있습니다.

또는 *{productname}*를 사용하여 운영 체제를 설치한 후 한 번에 *{productname}*에 등록할 수 있습니다. 이 방법은 부분적으로 또는 전체적으로 자동화되어 있으므로, 설치 프로그램 질문에 답하는 시간을 절약할 수 있으며 설치 및 등록하려는 클라이언트가 많은 경우 특히 유용합니다.

*{productname}*에서는 다음과 같이 여러 방법으로 운영 체제를 설치할 수 있습니다.

- \* 클라이언트가 이미 등록된 현재 위치에서 설치
- \* PXE 부팅을 사용하여 네트워크에서 설치
- \* 설치 CD-ROM 또는 USB 키를 준비한 후 시스템으로 이동하여 해당 미디어를 부팅하여 설치
- \* *{productname}* {smr} 솔루션의 일부로 설치

현재 위치 재설치 방법에서는 이전 운영 체제가 클라이언트에 이미 설치되어 있고 클라이언트가 이미 *{productname}*에 등록된 경우를 가정합니다.

현재 위치 설치 방법은 [xref:client-configuration:autoinst-reinstall.adoc](#)[등록된 시스템 재설치]를 참조하십시오.

네트워크 부팅 설치 방법은 포맷되지 않은 시스템에서 작동합니다. 그러나 다음과 같은 특정 네트워크 구성에서만 수행될 수 있습니다.

- \* *{productname}* 서버 또는 프록시 중 하나가 설치하는 시스템과 동일한 로컬 네트워크에 있거나 그 사이의 모든 라우터를 통과하는 DHCP 릴레이가 있는 경우
- \* 새 DHCP 서버를 설정할 수 있거나 기존 DHCP 서버를 구성할 수 있는 경우
- \* 설치할 클라이언트가 PXE를 사용하여 부팅할 수 있거나 그렇게 구성할 수 있는 경우.

네트워크 부팅 방법은 [xref:client-configuration:autoinst-pxeboot.adoc](#)[네트워크를 통한 설치]를 참조하십시오.

이동식 미디어 방법을 사용하면 이러한 네트워크 제약을 우회할 수 있습니다. 그러나 이 경우에는 시스템이 CD-ROM 또는 USB 키를 읽어 부팅할 수 있는 것으로 가정합니다. 또한 클라이언트 시스템에 물리적으로도 액세스할 수 있어야 합니다.

이동식 미디어 방법은 [xref:client-configuration:autoinst-cdrom.adoc](#)[CD-ROM 또는 USB 키를 사용한 설치]를 참조하십시오.

*{productname}* {smr} 방법은 [xref:retail:retail-overview.adoc](#)[리테일 가이드]를

참조하십시오.

[ NOTE ]

=====

{ubuntu} 및 {debian} 클라이언트의 자동 설치는 지원되지 않습니다. 이러한 운영 체제는 수동으로 설치해야 합니다.

=====

{productname}의 자동 설치 기능은 Cobbler 소프트웨어를 기반으로 합니다. Cobbler에 대한 자세한 설명은 <https://cobbler.github.io/quickstart/>를 참조하십시오.

[ NOTE ]

=====

{suse}는 {productname} {webui}에서 또는 {productname} API를 통해 사용할 수 있는 Cobbler 함수만 지원합니다. 여기에는 지원되는 기능에 대한 설명만 제공됩니다.

=====

:leveloffset: 3

:leveloffset: +2

[ [autoinst-reinstall] ]

= 등록된 시스템 재설치

현재 위치 다시 설치는 로컬 클라이언트 시스템에서 시작됩니다. 따라서 PXE를 사용하여 네트워크에서 클라이언트를 부팅할 필요가 없습니다.

등록된 클라이언트를 현재 위치에 다시 설치하려면 자동 설치 가능한 배포판 및 자동 설치 프로파일을 정의해야 합니다. 관련 정보는 [xref:client-configuration:autoinst-distributions.adoc](#)[자동 설치 가능한 배포판] 및 [xref:client-configuration:autoinst-profiles.adoc](#)[자동 설치 프로파일]을 참조하십시오.

자동 설치 프로파일 및 배포판을 정의한 후 다시 설치를 실행할 수 있습니다.

. 절차: 이미 등록된 클라이언트 다시 설치

. {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 다시 설치할 클라이언트를 선택한 후 menu:프로비저닝[자동 설치 > 일정] 하위 탭으로 이동합니다.

. 준비한 자동 설치 프로파일을 선택하고 필요한 경우 프록시를 선택한 후 btn:[자동 설치 스케줄링 후 완료]를 클릭합니다.

. 클라이언트가 기존 클라이언트이고 osad를 구성하지 않은 경우에는 작업을 가져올 때까지 기다려야 합니다.

. menu:프로비저닝[자동 설치 > 세션 상태]로 이동하거나 클라이언트에서 직접 설치 진행 상황을 모니터링할 수 있습니다. 클라이언트가 다시 부팅되고 부팅 메뉴에서 이름이 [guimenu]``reinstall-system``인 새 항목을 선택합니다.

image::autoinstall\_reinstall.png[scaledwidth=60%]

그리면 HTTP 프로토콜을 통해 설치가 진행됩니다.

```
:leveloffset: 3
:leveloffset: +2

[[autoinst-pxeboot]]
= 네트워크를 통한 설치(PXE 부팅)
```

네트워크 부팅 설치 중:

- . 클라이언트는 PXE 모드로 부팅됩니다.
- . DHCP 서버는 해당 서버의 IP 주소 및 마스크, 설치 서버의 주소 및 부트로더 파일의 이름을 클라이언트에 제공합니다.
- . 클라이언트는 설치 서버에서 TFTP 프로토콜을 통해 부트로더 파일을 다운로드한 후 실행합니다.
- . 클라이언트에는 메뉴에서 설치할 수 있는 프로파일을 선택하거나 프로파일 중 하나의 자동 설치를 시작할 수 있는 옵션이 제공됩니다.
- . 클라이언트는 해당 프로파일과 일치하는 배포판의 커널 및 초기 RAM 디스크를 TFTP 프로토콜을 통해 다운로드합니다.
- . 설치 커널은 {kickstart} 또는 {ay} 설치 프로그램을 시작합니다. 이제부터 HTTP 프로토콜을 통해 서버에 제공되는 리소스가 사용됩니다.
- . {kickstart} 또는 {ay} 프로파일에 따라 배포판이 자동으로 설치됩니다.
- . 프로파일은 클라이언트를 {productname} 서버에 기존 또는 Salt 클라이언트로 등록하는 코드 조각을 호출합니다.

`image::cobbler_menu.png[scaledwidth=100%]`

설치 서버는 {productname} 서버 또는 프록시 중 하나일 수 있습니다. 프록시에서 설치하려면 서버와 프록시 간에 TFTP 트리를 동기화한 후 시작해야 합니다.

DHCP 서버는 호스트 이름, 라우터의 주소 및 도메인 이름 서버의 주소와 같은 기타 구성 정보를 클라이언트에 제공할 수도 있습니다. 이러한 정보 중 일부는 예를 들어 도메인 이름으로 설치 서버를 지정하는 경우 자동 설치에 필요할 수 있습니다.

PXE 부팅 메뉴에서 먼저 [guimenu] ``로컬 부팅``을 선택합니다. 이 옵션을 선택하면 로컬 디스크 드라이브에서 부팅 프로세스가 진행됩니다. 일정 시간이 지난 후 프로파일을 선택하지 않으면 이 옵션이 자동으로 선택됩니다. 이는 프로파일 중 하나를 선택하는 사용자 작업이 없는 경우 자동 설치가 시작되지 않도록 하는 보안 조치입니다.

또는 수동 개입 없이 프로파일 중 하나에서 설치가 자동으로 시작될 수 있습니다. 이를 "무인 프로비저닝"이라고 합니다.

"베어메탈" 기능은 PXE 부팅 기반의 무인 프로비저닝 중 하나입니다. 이 경우에는 부트로더 파일이 {productname} 서버에서 클라이언트 등록만을 수행하며 설치가 시작되지 않습니다. 그러면 현재 위치에서 나중에 다시 설치할 수 있습니다.

- . 절차: PXE 부팅으로 설치
- . DHCP 서버를 준비합니다. 관련 내용은 [xref:client-configuration:autoinst-pxeboot.adoc#prepare-the-dhcp-server](#)[DHCP 서버 준비]를 참조하십시오.
- . 자동 설치 가능한 배포판을 준비합니다. 관련 내용은 [xref:client-configuration:autoinst-distributions.adoc](#)[자동 설치 가능한 배포판]을 참조하십시오.
- . 자동 설치 프로파일을 준비합니다. 관련 내용은 [xref:client-configuration:autoinst-profiles.adoc](#)[자동 설치 프로파일]을 참조하십시오.
- . 클라이언트를 다시 부팅하고 설치할 프로파일을 선택합니다.

일부 다른 단계는 선택 사항입니다. 프록시를 설치 서버로 사용하려면 [xref:client-configuration:autoinst-pxeboot.adoc#synchronize-the-tftp-tree-with-proxies](#)[프록시와 TFTP 트리 동기화]를 참조하십시오. 무인 프로비저닝에 대한 설명은 [xref:client-configuration:autoinst-unattended.adoc](#)[무인 프로비저닝]을 참조하십시오.

**[[prepare-the-dhcp-server]]**  
== DHCP 서버 준비

PXE 부팅 프로세스는 DHCP를 사용하여 TFTP 서버를 찾습니다. {productname} 서버 또는 프록시가 TFTP 서버와 같은 역할을 할 수 있습니다.

네트워크의 DHCP 서버에 대한 관리자 액세스 권한이 있어야 합니다. 설치 서버를 TFTP 부팅 서버로 가리키도록 DHCP 구성 파일을 편집합니다.

- . 예: ISC DHCP 서버 구성
- . DHCP 서버에서 루트 권한으로 [path]``/etc/dhcpd.conf`` 파일을 엽니다.
- . 클라이언트의 선언을 수정합니다.

```
host myclient { (...)
 next-server 192.168.2.1;
 파일 이름 "pxelinux.0"; }
```

- . 파일을 저장하고 [systemitem]``dhcpd`` 서비스를 다시 시작합니다.

이 예에서는 PXE 클라이언트인 ``myclient``를 ``192.168.2.1``의 설치 서버로 지정하고 [path]``pxelinux.0`` 부트로더 파일을 검색하도록 지시합니다.

또는 DHCP 서버가 {productname}에 등록된 경우 DHCPd 수식을 대신 사용하여 구성할 수 있습니다.

- . 예: DHCPd 수식을 사용한 ISC DHCP 서버 구성
- . menu: 시스템[시스템 목록]으로 이동하여 변경할 클라이언트를 선택한 후 [guimenu]``수식``탭으로 이동하여 DHCPd 수식을 활성화합니다.
- . 수식의 [guimenu]``Dhcpd`` 탭으로 이동하여 [guimenu]``다음 서버`` 필드에 설치 서버의 호스트 이름 또는 IP 주소를 입력합니다.
- . [guimenu]``파일 이름 EFI`` 필드에 [path]``grub/grub.efd``를 입력하여 EFI PXE 지원을 활성화합니다.
- . [guimenu]``파일 이름`` 필드에 [path]``pxelinux.0``을 입력하여 레거시 BIOS 지원을 활성화합니다.
- . btn:[수식 저장]을 클릭하여 구성을 저장합니다.
- . highstate를 적용합니다.

[NOTE]

=====

보안 부팅을 사용하는 경우 [guimenu]``Filename EFI`` 필드에 [path]``grub/shim.efd`` 대신 [path]``grub/shim.efd``를 입력하십시오.

=====

이렇게 하면 모든 호스트에 대한 전역 PXE 서버가 설정되고 사용자를 호스트별로 설정할 수 있습니다. DHCPd 수식에 대한 자세한 설명은 [xref:specialized-guides:salt/salt-formula-dhcpd.adoc](#)[DHCPd 수식]을 참조하십시오.

[[ synchronize-the-tftp-tree-with-proxies ]]

== TFTP 트리와 프록시 동기화

{productname} 서버의 TFTP 트리와 {productname} 프록시를 동기화할 수 있습니다. 동기화하려면 HTTPS 포트 443을 열어야 합니다.

[WARNING]

=====

추가된 모든 프록시는 트리 동기화의 속도를 느리게 합니다.

=====

. 절차: 서버와 프록시 간에 TFTP 동기화

- . {productname} 서버의 명령 프롬프트에서 루트 권한으로 [systemitem]``susemanager-tftpsync`` 패키지를 설치합니다.

`zypper install susemanager-tftpsync`

- . {productname} 프록시의 명령 프롬프트에서 루트 권한으로 [systemitem]``susemanager-tftpsync-recv`` 패키지를 설치합니다.

```
zypper install susemanager-tftpsync-recv
```

- . 프록시에서 루트 권한으로 [command]``configure-tftpsync.sh`` 스크립트를 실행합니다. 스크립트는 대화식으로 {productname} 서버 및 프록시의 호스트 이름 및 IP 주소에 대한 세부 정보와 함께 프록시의 [path]``tftpboot`` 디렉토리 위치를 묻습니다. 자세한 내용을 확인하려면 [command]``configure-tftpsync.sh --help`` 명령을 사용하십시오.
  - . 서버에서 루트 권한으로 [command]``configure-tftpsync.sh`` 스크립트를 실행합니다.
- +

```
configure-tftpsync.sh proxy1.example.com proxy2.example.com
```

- . 서버에서 [command]``cobbler sync`` 명령을 실행하여 파일을 프록시로 푸시합니다. 프록시를 올바르게 구성하지 않으면 이 작업이 실패합니다.

나중에 프록시 목록을 변경하려면 [command]``configure-tftpsync.sh`` 스크립트를 사용해 편집할 수 있습니다.

#### [NOTE]

=====

이미 구성된 프록시를 다시 설치하고 모든 파일을 다시 푸시하려면 [command]``cobbler sync``를 호출하기 전에 [path]``/var/lib/cobbler/pxe\_cache.json``에서 캐시 파일을 제거해야 합니다.

=====

```
:leveloffset: 3
:leveloffset: +2
```

#### [[autoinst-cdrom]]

= CD-ROM 또는 USB 키를 통한 설치

{productname}에 등록되지 않고 PXE를 통한 네트워크 부팅을 사용할 수 없는 클라이언트의 경우 부팅 가능한 CD-ROM 또는 USB 키를 사용하여 시스템을 설치할 수 있습니다.

해당 이동식 매체를 준비하기 위한 한 가지 옵션은 Cobbler를 사용하는 것입니다. Cobbler를 사용한 ISO 이미지 준비에 대한 설명은 [xref:client-configuration:autoinst-cdrom.adoc#build-iso-with-cobbler](#)[Cobbler를 사용한 ISO 이미지 빌드]를 참조하십시오.

다른 옵션은 배포판에 따른 방식을 사용하는 것입니다.

\* {suse} 시스템의 경우 KIWI를 사용하여 ISO 이미지를 준비합니다. 관련 정보는 [xref:client-configuration:autoinst-cdrom.adoc#build-iso-with-kiwi](#)[KIWI를 사용한 SUSE ISO 이미지 빌드]를 참조하십시오.

\* {redhat} 시스템의 경우 ``mkisofs``를 사용합니다. 관련 정보는 [xref:client-configuration:autoinst-cdrom.adoc#build-iso-with-mkisofs](#)[mkisofs를 사용한 RedHat ISO 이미지 빌드]를 참조하십시오.

모든 경우에 결과 이미지를 CD-ROM으로 굽거나 USB 키를 준비해야 합니다.

**[[build-iso-with-cobbler]]**

== Cobbler를 사용한 ISO 이미지 빌드

Cobbler는 PXE 설치와 유사한 방식으로 작동하는 배포 세트, 커널 및 메뉴가 포함된 ISO 부팅 이미지를 생성할 수 있습니다.

[NOTE]

====

{ibmz}에서는 Cobbler를 이용한 ISO 빌드를 지원하지 않습니다.

====

Cobbler를 사용하여 ISO 이미지를 준비하려면 PXE를 통한 네트워크 부팅을 사용하는 것과 유사하게 배포 및 프로파일을 준비해야 합니다. 배포판 생성에 대한 설명은 [xref:client-configuration:autoinst-distributions.adoc](#)[자동 설치 가능한 배포판]을 참조하십시오. 프로파일 생성에 대한 설명은 [xref:client-configuration:autoinst-profiles.adoc](#)[자동 설치 프로파일]을 참조하십시오.

The Cobbler [command]``buildiso`` command takes parameters to define the name and output location of the boot ISO. Specifying the distribution with [option]``--distro`` is mandatory when running [command]``buildiso`` command.

```
cobbler buildiso --iso=/path/to/boot.iso --distro=<your-distro-label>
```

[IMPORTANT]

====

You must use distro and profile labels as listed by Cobbler, and not simply as shown in the UI.

====

To list the names of distributions and profiles stored by Cobbler, run the commands:

## 4.10.2. cobbler distro list

### 4.10.3. cobbler profile list

부팅 ISO에는 모든 프로파일 및 시스템이 기본적으로 포함되어 있습니다. [option]` `--profiles`` 및 [option]` `--systems`` 옵션을 사용하면 사용되는 프로파일 및 시스템을 제한할 수 있습니다. 예:

```
cobbler buildiso --systems="system1 system2 system3" \ --profiles=" <your-profile2-label> <your-profile3-label>" --distro=<your-distro-label>"
```

[NOTE]

=====

ISO 이미지를 공용 [path]` `tmp`` 디렉토리에 쓸 수 없는 경우  
[path]` `/usr/lib/systemd/system/cobblerd.service``에서 systemd 설정을  
확인하십시오.

=====

[ [build-iso-with-kiwi]]

== KIWI를 사용한 SUSE ISO 이미지 빌드

KIWI는 이미지 생성 시스템입니다. KIWI를 사용하여 {suse} 시스템을 설치하기 위한 대상 시스템에서 사용되는 부팅 가능한 ISO 이미지를 생성할 수 있습니다. 시스템은 재부팅되거나 켜질 때 이 이미지로 부팅되고, {productname}에서 {ay} 구성 파일을 로드하며, {ay} 프로파일에 따라 {sles}를 설치합니다.

ISO 이미지를 사용하려면 시스템을 부팅하고 프롬프트에 ``autoyast``를 입력하십시오({ay} 부팅을 위한 레이블을 ``autoyast`` 상태로 두었다고 가정함). kbd:[Enter] 키를 눌러 {ay} 설치를 시작합니다.

////

we would love a bit more details - ebischoff

////

KIWI에 대한 자세한 정보는 <http://doc.opensuse.org/projects/kiwi/doc/>에서 참조하십시오.

[ [build-iso-with-mkisofs]]

== mkisofs를 사용한 RedHat ISO 이미지 빌드

[command]` `mkisofs``를 사용하여 {redhat} 시스템을 설치하기 위한 대상 시스템에서 사용되는 부팅 가능한 ISO 이미지를 생성할 수 있습니다. 시스템은 재부팅되거나 켜질 때 이 이미지로 부팅되고, {productname}에서 {kickstart} 구성 파일을 로드하며, {kickstart} 프로파일에 따라 {rhel}를 설치합니다.

- . 절차: mkisofs를 사용한 부팅 가능한 ISO 빌드
- . 대상 배포의 첫 CD-ROM에서 [path]``/isolinux``의 내용을 복사합니다.
- . [path]``isolinux.cfg`` 파일을 편집하여 기본값을 'ks'로 설정합니다. 'ks' 섹션을 다음과 같이 변경합니다.

+

```
label ks kernel vmlinuz append text ks=url initrd=initrd.img lang= devfs=nomount \ ramdisk_size=16438
ksdevice
```

+

IP 주소 기반 {kickstart} URL은 다음과 같습니다.

+

[http://my.manager.server/kickstart/ks/mode/ip\\_range](http://my.manager.server/kickstart/ks/mode/ip_range)

+

IP 범위를 통해 정의되는 {kickstart} 배포는 오류 발생을 방지하기 위해 현재 빌드 중인 것에서 배포하는 것과 일치해야 합니다.

- . 옵션: [replaceable]``ksdevice``를 사용하려면 다음과 같아야 합니다.

+

ksdevice=eth0

+

새로운 배포 레이블을 지정하여 패밀리 내에서 Kickstart 프로파일에 대한 배포를 변경할 수 있습니다(예: {rhel} AS 4에서 {rhel} ES 4로 변경). 버전 간(4에서 5로) 또는 업데이트 간(U1에서 U2로) 변경은 불가능하다는 점에 유의하십시오.

- . 필요에 따라 [path]``isolinux.cfg``를 보다 세부적으로 사용자 정의합니다. 예를 들어, 여러 옵션, 다양한 부팅 메시지 또는 더 짧은 시간 제한 기간을 추가할 수 있습니다.
- . 다음 명령으로 ISO를 생성합니다.

+

```
mkisofs -o file.iso -b isolinux.bin -c boot.cat -no-emul-boot \ -boot-load-size 4 -boot-info-table -R -J -v -T
isolinux/
```

+

[path]``isolinux/``는 배포 CD에서 복사되는 수정된 isolinux 파일을 포함하는 디렉토리의 상대 경로이고, [path]``file.iso``는 현재 디렉토리에 배치되는 출력 ISO 파일입니다.

- . ISO를 CD-ROM에 굽고 디스크를 삽입합니다. 또는 USB 키를 준비하여 삽입합니다.
- . 시스템을 부팅하고 프롬프트에서 [command]``ks``를 입력합니다(Kickstart 부팅에 대한

레이블을 ‘ks’로 둔 경우).

- . kbd:[Enter] 키를 눌러 {kickstart}를 시작합니다.

```
:leveloffset: 3
```

```
:leveloffset: +2
```

**[ [autoinst-distributions]]**

= 자동 설치 가능한 배포판

자동 설치 프로세스에서는 여러 파일을 사용하여 설치를 시작합니다. 이러한 파일에는 Linux 커널, 초기 RAM 디스크 및 설치 모드에서 운영 체제를 부팅하기 위해 필요한 기타 파일이 포함됩니다.

필요한 파일을 DVD 이미지에서 추출할 수 있습니다. 관련 정보는 [xref:client-configuration:autoinst-distributions.adoc#based-on-iso-image](#)[ ISO 이미지 기반 배포]를 참조하십시오.

또는 [package]``tftpboot-installation`` 패키지를 설치할 수 있습니다. 관련 정보는 [xref:client-configuration:autoinst-distributions.adoc#based-on-rpm-package](#)[RPM 패키지 기반 배포]를 참조하십시오.

또한 해당 파일과 운영 체제 버전이 동일한 {productname} 서버에서 동기화된 기본 채널이 있어야 합니다.

파일이 준비되고 기본 채널이 동기화되면 배포를 선언해야 합니다. 이 작업은 기본 채널에 설치 파일을 연결합니다. 배포는 1개 이상의 설치 프로파일에 의해 참조될 수 있습니다. 관련 정보는 [xref:client-configuration:autoinst-distributions.adoc#declare-distribution](#)[자동 설치 가능 배포판 선언]을 참조하십시오.

**[ [based-on-iso-image]]**

**== ISO 이미지 기반 배포**

이 방법에서는 클라이언트에 설치하려는 운영 체제의 설치 미디어가 있는 경우를 가정합니다. 일반적으로 Linux 커널, [path]``initrd`` 파일 및 설치 모드에서 운영 체제를 부팅하는 데 필요한 기타 파일을 포함하는 DVD [path]``.iso`` 이미지입니다.

. 절차: 설치 미디어에서 파일 임포트

- . {productname} 서버에 설치 미디어를 복사합니다. {suse} 운영 체제의 경우 <https://www.suse.com/download/>에서 설치 미디어를 다운로드할 수 있습니다.

- . ISO 이미지를 루프 마운트하고 컨텐트를 원하는 위치에 복사합니다.

+

#### 4.10.4. mount -o loop,ro <image\_name>.iso /mnt

#### 4.10.5. mkdir -p /srv/www/distributions

#### 4.10.6. cp -a /mnt /srv/www/distributions/<image\_name>

#### 4.10.7. umount /mnt

+

- . Take a note of the file path. You will need it when you declare the distribution to {productname}.

[ [based-on-rpm-package] ]

== RPM 패키지 기반 배포

이 방법은 {suse} 시스템에 적용됩니다. 설치 시스템에서 미리 패키지화된 파일을 사용하므로 설치 미디어에서 컨텐트를 임포트하는 것보다 단순합니다.

. 절차: 설치 패키지에서 파일 추출

- . {productname} 서버에서 이름이 [package]``tftpboot-installation``으로 시작하는 패키지를 설치합니다. 정확한 이름은 [command]``zypper se tftpboot-installation`` 명령으로 확인할 수 있습니다.
- . [command]``ls -d /usr/share/tftpboot-installation/\*`` 명령으로 설치 파일의 위치를 확인합니다. 파일 경로를 메모해 둡니다. {productname}에 대한 배포를 선언할 때 필요합니다.

이 절차를 통해 {productname} 서버와 버전이 동일한 운영 체제를 설치하도록 준비할 수 있습니다. 클라이언트에 다른 운영 체제 또는 버전을 설치하려면 해당 배포에서 [package]``tftpboot-installation-\*`` 패키지를 수동으로 가져와야 합니다. {productname}의 [menu]``패키지 검색`` 입력 상자에서 이름이 [package]``tftpboot-installation``으로 시작하는 패키지를 검색한 후 패키지의 세부 정보를 확인합니다. 해당 정보는 [path]``/var/spacewalk`` 아래의 로컬 경로에 표시됩니다.

[ [declare-distribution] ]

== 자동 설치 가능한 배포판 선언

자동 설치 파일을 추출한 후의 단계는 자동 설치 가능한 배포판을 선언하는 것입니다.

. 절차: 자동 설치 가능한 배포판 선언

- . {productname} {webui}에서 menu:시스템[자동 설치 > 배포]로 이동합니다.
- . [guimenu]``배포 생성``을 클릭하고, 다음 필드를 채웁니다.

- +
  - \* [guimenu] ``배포 레이블`` 필드에서 자동 설치 가능한 배포를 식별할 수 있는 이름을 입력합니다.
  - \* [guimenu] ``트리 경로`` 필드에서 {productname} 서버에 저장된 설치 미디어의 경로를 입력합니다.
  - \* 일치하는 [guimenu] ``기본 채널`` 을 선택합니다. 설치 미디어와 일치해야 합니다.
  - \* [guimenu] ``설치 프로그램 생성`` 을 선택합니다. 설치 미디어와 일치해야 합니다.
  - \* 선택 사항: 이 배포를 부팅할 때 사용할 커널 옵션을 지정하십시오. 여러 가지 방식으로 커널 옵션을 제공할 수 있습니다. 배포에 일반적으로 사용되는 옵션만 여기에 추가하십시오.
  - . btn:[자동 설치 가능한 배포 생성]을 클릭합니다.

준비한 설치 파일에는 설치에 필요한 패키지가 포함되지 않을 수 있습니다. 포함되지 않은 경우, [option] ``useonlinerepo=1`` 을 [guimenu] ``커널 옵션`` 필드에 추가합니다.

패키지 리포지토리에는 서명되지 않을 수 있는 메타데이터가 포함되어 있습니다. 메타데이터가 서명되지 않은 경우 [option] ``insecure=1`` 을 [guimenu] ``커널 옵션`` 필드에 추가하거나 xref:client-configuration:autoinst-ownpgkey.adoc[자체 GPG 키 사용]에서의 설명과 같이 자체 GPG 키를 사용합니다.

이러한 커널 옵션은 예를 들어 전체 DVD가 아닌 "online installer" ISO 이미지를 사용하거나 [package] ``tpboot-installation`` 패키지를 사용할 때 필요합니다.

menu:시스템[자동 설치 > 배포판]으로 이동하여 자동 설치 가능한 배포판을 관리합니다.

```
:leveloffset: 3
:leveloffset: +2
```

#### [[autoinst-profiles]]

= 자동 설치 프로파일

운영 체제가 설치되는 방법은 자동 설치 프로파일에 따라 다릅니다. 예를 들어, 설치 프로그램에 전달한 추가 커널 파라미터를 지정할 수 있습니다.

프로파일에서 가장 중요한 부분은 "자동 설치 파일"입니다. 수동으로 설치하는 경우 파티셔닝 및 네트워크 정보, 사용자 상세 정보와 같은 정보를 설치 프로그램에 제공해야 합니다. 자동 설치 파일은 이러한 정보를 스크립트된 형식으로 제공하는 방법입니다. 이러한 유형의 파일을 가끔 "답변 파일"이라고도 합니다.

{productname}에서 설치하려는 클라이언트의 운영 체제에 따라 두 가지 유형의 프로파일을 사용할 수 있습니다.

- \* {sle} 또는 {opensuse} 클라이언트에 대해서는 {ay}를 사용하십시오.
- \* {rhel} 클라이언트에 대해서는 {kickstart}를 사용하십시오.

운영 체제가 다양한 클라이언트를 설치하려는 경우 {ay} 및 {kickstart} 프로파일을 모두 사용할 수 있습니다.

- \* 프로파일을 선언하는 방법은 [xref:client-configuration:autoinst-profiles.adoc#declare-profile](#)[프로파일 선언]을 참조하십시오.
- \* {ay} 프로파일에 대한 설명은 [xref:client-configuration:autoinst-profiles.adoc#autoyast](#)[AutoYaST 프로파일]을 참조하십시오.
- \* {kickstart} 프로파일에 대한 설명은 [xref:client-configuration:autoinst-profiles.adoc#kickstart](#)[Kickstart 프로파일]을 참조하십시오.

프로파일에 포함된 자동 설치 파일에는 변수 및 코드 조각이 포함될 수 있습니다. 변수 및 코드 조각에 대한 설명은 [xref:client-configuration:autoinst-profiles.adoc#templates-syntax](#)[템플릿 구문]을 참조하십시오.

**[[declare-profile]]**

== 프로파일 선언

자동 설치 파일 및 배포를 준비했으면 프로파일을 생성하여 {productname} 서버에서 자동 설치를 관리할 수 있습니다. 프로파일은 선택한 이 배포판을 설치하는 방법을 결정합니다. 프로파일을 만드는 한 가지 방법은 {ay} 또는 {kickstart} 파일을 업로드하는 것입니다. 또는 {kickstart}의 경우에만 {webui} 마법사를 사용할 수 있습니다.

. 절차: 업로드하여 자동 설치 프로파일 생성

- . {productname} {webui}에서 menu:시스템[자동 설치 > 프로파일]로 이동합니다.
- . btn:[Kickstart/Autoyast 파일 업로드]를 클릭합니다.
- . [guimenu]``레이블`` 필드에 프로파일의 이름을 입력합니다. 공백을 사용하면 안 됩니다.
- . [guimenu]``자동 설치 트리`` 필드에서 이 프로파일에 사용할 자동 설치 가능한 배포판을 선택합니다.
- . [guimenu]``가상화 유형`` 필드에서 이 프로파일에 사용할 가상화 유형을 선택하거나, 이 프로파일을 사용하여 새로운 가상 시스템을 생성하지 않으려면 ``없음``을 선택합니다.
- . 자동 설치 파일의 내용을 [guimenu]``파일 내용`` 필드에 복사하거나 [guimenu]``업로드할 파일`` 필드를 사용하여 직접 파일을 업로드합니다.

+

For more information about the details to include here, see [xref:client-configuration:autoinst-profiles.adoc#autoyast](#)[AutoYast Profiles] or [xref:client-configuration:autoinst-profiles.adoc#kickstart](#)[Kickstart Profiles].

- . btn:[생성]을 클릭하여 프로파일을 생성합니다.

. 절차: 마법사로 {kickstart} 프로파일 생성

- . {productname} {webui}에서 menu:시스템[자동 설치 > 프로파일]로 이동합니다.
- . btn:[Kickstart 프로파일 생성]을 클릭합니다.
- . [guimenu] ``레이블`` 필드에 프로파일의 이름을 입력합니다. 공백을 사용하면 안 됩니다.
- . [guimenu] ``기본 채널`` 필드에서 이 프로파일에 사용할 기본 채널을 선택합니다. 이 필드는 사용 가능한 배포에서 채워집니다. 필요한 기본 채널을 사용할 수 없는 경우 배포를 올바르게 생성하였는지 확인하십시오.
- . [guimenu] ``가상화 유형`` 필드에서 이 프로파일에 사용할 가상화의 유형을 선택합니다. 가상화가 없는 경우 ``없음``을 선택합니다.
- . btn:[다음]을 클릭합니다.
- . [guimenu] ``배포 파일 위치``에서 {productname} 서버에 설치된 설치 미디어의 경로를 입력합니다.
- . btn:[다음]을 클릭합니다.
- . 클라이언트에서 루트 사용자에게 비밀번호를 제공합니다.
- . btn:[완료]를 클릭합니다.
- . 새 프로파일의 상세 정보를 검토하고 필요에 따라 사용자 정의합니다.

자동 설치 프로파일을 생성할 때 [guimenu] ``이 기본 채널에 항상 최신 트리를 사용합니다``의 확인란을 선택할 수 있습니다. 이 설정은 {productname}가 특정 기본 채널과 연결된 최신 배포를 자동으로 선택하도록 허용합니다. 나중에 새 배포를 추가하는 경우 {productname}는 최근에 생성되거나 수정된 것을 사용합니다.

[guimenu] `` 가상화 유형``을 변경하려면 일반적으로 프로파일 부트로더 및 파티션 옵션을 변경해야 합니다. 이렇게 하면 사용자 정의가 덮어쓰기될 수 있습니다. 신규 또는 변경된 설정을 저장하기 전에 [guimenu] ``파티셔닝`` 탭으로 이동하여 확인하십시오.

배포판 및 프로파일의 커널 옵션은 결합되어 있습니다.

menu:시스템[자동 설치 > 프로파일]로 이동해 편집하려는 프로파일의 이름을 클릭하여 자동 설치 프로파일의 상세 정보 및 설정을 변경할 수 있습니다. 또는 menu:시스템[시스템 목록]으로 이동하여 프로비저닝할 클라이언트를 선택한 후 menu:프로비저닝[자동 설치] 하위 탭으로 이동합니다.

```
[[autoyast]]
== AutoYast 프로파일
```

{ay} 프로파일은 프로파일을 식별하는 [guimenu]``레이블``, 자동 설치 가능한 배포판을 가리키는 [guimenu]``자동 설치 트리``, 여러 옵션과 가장 중요한 {ay} 설치 파일로 구성됩니다.

{ay} 설치 파일은 {ay} 설치 프로그램에 방향을 제공하는 XML 파일입니다. {ay}에서는 이 파일을 "제어 파일"이라고 부릅니다. {ay} 설치 파일의 전체 구문은 [https://doc.opensuse.org/projects/autoyast/#cha-configuration-installation-options\[\]](https://doc.opensuse.org/projects/autoyast/#cha-configuration-installation-options[])를 참조하십시오.

{suse}는 사용자 정의 파일의 시작점으로 사용할 수 있는 {ay} 설치 파일의 템플릿을 제공합니다. 이 템플릿은 [path]``AutoYast`` 디렉토리의 [https://github.com/SUSE/manager-build-profiles\[\]](https://github.com/SUSE/manager-build-profiles[])에서 확인할 수 있습니다. 이러한 각 프로파일은 사용하기 전 일부 변수를 설정해야 합니다. 필요한 변수는 스크립트에 포함된 [path]``README`` 파일에서 확인합니다. {ay} 스크립트에서 변수 사용에 대한 자세한 설명은 [xref:client-configuration:autoinst-profiles#variables\[변수\]](#)를 참조하십시오.

다음은 {productname}(으)로 설치하기 위한 {ay} 설치 파일에서 가장 중요한 부분입니다.

\* ``<add-on>``은 설치에 하위 채널을 추가할 수 있습니다.

+

``<add-on>`` 예제가 있는

[https://doc.opensuse.org/projects/autoyast/#Software-Selections-additional\[\]](https://doc.opensuse.org/projects/autoyast/#Software-Selections-additional[])에서 참조하십시오.

\* ``<general>\$SNIPPET('spacewalk/sles\_no\_signature\_checks')</general>``은 서명 확인을 비활성화합니다.

\* ``<software>``를 사용하면 {unifiedinstaller}에 제품을 지정할 수 있습니다.

+

"<software>" 예제가 있는

[https://doc.opensuse.org/projects/autoyast/#CreateProfile-Software\[\]](https://doc.opensuse.org/projects/autoyast/#CreateProfile-Software[])를 참조하십시오.

\* ``<init-scripts config:type="list">\$SNIPPET('spacewalk/minion\_script')</init-scripts>`` 을 사용하면 클라이언트가 {productname}를 Salt client로 등록할 수 있습니다.

{ay}에 대한 자세한 내용은 [https://doc.opensuse.org/projects/autoyast/\[\]](https://doc.opensuse.org/projects/autoyast/[])를 참조하십시오.

AutoYast를 대체하는 가장 최신 Salt 기반 제품은 Yomi입니다. Yomi에 대한 설명은 [xref:specialized-guides:salt/salt-yomi.adoc\[Yomi를 사용한 설치\]](#)를 참조하십시오.

[[kickstart]]

## == Kickstart 프로파일

{kickstart} 프로파일은 매우 다양한 구성 옵션을 제공합니다. 이러한 프로파일을 생성하려면 해당 프로파일을 업로드하거나 전용 마법사를 사용합니다.

{kickstart} 프로파일을 사용하면 파일 보관 목록을 사용할 수 있습니다. {kickstart}를 사용하여 다시 설치할 클라이언트에 사용자 정의 구성 파일이 여러 개 있는 경우 해당 파일을 목록으로 저장하고 사용할 {kickstart} 프로파일과 이 목록을 연결할 수 있습니다.

. 절차: 파일 보관 목록 생성

- . {productname} {webui}에서 menu:시스템[자동 설치 > 파일 보관]으로 이동하여 btn:[파일 유지 목록 생성]을 클릭합니다.
- . 적절한 레이블을 입력하고 저장하려는 모든 파일 및 디렉토리의 절대 경로를 나열합니다.
- . btn:[목록 생성]을 클릭합니다.
- . {kickstart} 프로파일에 파일 보관 목록 포함
  - . menu:시스템[자동 설치 > 프로파일]로 이동한 후 편집할 프로파일을 선택하고 menu:시스템 세부 정보[파일 보관] 하위 탭으로 이동하여 포함할 파일 보관 목록을 선택합니다.

### [ NOTE ]

=====

파일 유지 목록은 전체 크기가 1{nbsp}MB로 제한됩니다. [path]``/dev/hda1`` 및 [path]``/dev/sda1``과 같은 특수 장치는 보관할 수 없습니다. 파일 및 디렉토리 이름만 사용하십시오. 정규식 와일드카드는 사용할 수 없습니다.

=====

Kickstart에 대한 자세한 설명은 Red Hat 문서를 참조하십시오.

## [[templates-syntax]]

### == 템플릿 구문

설치 파일의 일부는 설치 중에 대체됩니다. 변수는 단일 값으로 대체되며 코드 조각은 텍스트의 전체 섹션으로 대체됩니다. 이스케이프된 기호 또는 섹션은 대체되지 않습니다.

Cheetah라는 템플릿 엔진을 사용하면 Cobbler가 해당 대체를 수행할 수 있습니다. 이 방식을 통해 각각에 대한 프로파일을 수동으로 생성할 필요 없이 대량의 시스템을 다시 설치할 수 있습니다.

{productname} {webui}에서 자동 설치 변수 및 코드 조각을 생성할 수 있습니다. 프로파일의 [guimenu]``자동 설치 파일`` 탭을 사용하면 대체 결과를 확인할 수 있습니다.

- \* 변수에 대한 설명은 [xref:client-configuration:autoinst-profiles#variables](#)[변수]를 참조하십시오.
- \* 코드 조각에 대한 설명은 [xref:client-configuration:autoinst-profiles#code-snippets](#)[코드 조각]을 참조하십시오.
- \* 이스케이프 기호 또는 전체 섹션에 대한 설명은 [xref:client-configuration:autoinst-profiles#variables](#)[이스케이핑]을 참조하십시오.

#### [[variables]]

==== 변수

자동 설치 변수를 사용해 값을 {kickstart} 및 {ay} 프로파일로 대신할 수 있습니다. 변수를 정의하려면 프로파일에서 [guimenu]``변수`` 하위 탭으로 이동하여 텍스트 상자에서 [replaceable]``name=value`` 쌍을 생성합니다.

예를 들어, 클라이언트의 IP 주소를 가진 변수와 게이트웨이의 주소를 가진 변수를 생성할 수 있습니다. 그러면 이러한 변수를 동일한 프로파일에서 설치된 모든 클라이언트에 대해 정의할 수 있습니다. 이 작업을 수행하려면 [guimenu]``변수`` 텍스트 상자에 다음 줄을 추가합니다.

```
ipaddr=192.168.0.28 gateway=192.168.0.1
```

변수를 사용하려면 프로파일에서 [option]``\$`` 기호를 앞에 추가하여 변수를 대체합니다. 예를 들어, {kickstart} 파일의 [option]``네트워크`` 부분은 다음과 같습니다.

```
network --bootproto=static --device=eth0 --onboot=on --ip=$ipaddr \ --gateway=$gateway
```

[option]``\$ipaddr``은 ``192.168.0.28``로, [option]``\$gateway``는 ``192.168.0.1``로 확인됩니다.

설치 파일에서 변수는 계층 구조를 사용합니다. 시스템 변수는 프로파일 변수에 우선하고, 프로파일 변수는 배포 변수에 우선합니다.

#### [[code-snippets]]

==== 코드 조각

{productname}는 미리 정의된 다양한 코드 조각과 함께 제공됩니다. [menu:시스템](#)[자동 설치 > 자동 설치 조각]으로 이동하여 기존 조각의 목록을 확인합니다.

자동 설치 파일에 [option]``\$SNIPPET()`` 매크로를 삽입하여 코드 조각을 사용합니다. 예를 들어 {kickstart}에서는 다음과 같습니다.

```
$SNIPPET('spacewalk/rhel_register_script')
```

또는 {ay}에서는 다음과 같습니다.

```
<init-scripts config:type="list"> $SNIPPET('spacewalk/sles_register_script') </init-scripts>
```

매크로는 Cobbler로 구문 분석되고 코드 조각의 컨텐트로 대체됩니다. 또한 나중에 자동 설치 파일에서 사용할 자체 코드 조각을 저장할 수 있습니다. btn:[조각 생성]을 클릭하여 새 코드 조각을 생성합니다.

다음 예에서는 일반 하드 드라이브 파티션 구성에 대한 {kickstart} 코드 조각을 다음과 같이 설정합니다.

```
clearpart --all part /boot --fstype ext3 --size=150 --asprimary part / --fstype ext3 --size=40000 --asprimary
part swap --recommended
```

```
part pv.00 --size=1 --grow
```

```
volgroup vg00 pv.00 logvol /var --name=var vgname=vg00 --fstype ext3 --size=5000
```

예를 들어 다음과 같이 코드 조각을 사용합니다.

```
$SNIPPET('my_partition')
```

```
[[escaping]]
== 이스케이핑
```

자동 설치 파일에 ``\$(example)``와(과) 같은 셸 스크립트 변수가 포함된 경우 백슬래시를 사용하여 컨텐트를 이스케이프해야 합니다. ``\\$`` 기호를 이스케이프하면 템플릿 엔진이 기호를 내부 변수로 평가하지 못하게 됩니다.

긴 스크립트 또는 문자열은 ``\#raw`` 및 ``\#end`` 지시어로 래핑하여 이스케이프할 수 있습니다. 예:

```
#raw #!/bin/bash for i in {0..2}; do echo "$i - Hello World!" done #end raw
```

``#`` 기호 다음에 공백이 있는 모든 줄은 주석으로 취급되므로 평가되지 않습니다. 예:

## 4.10.8. 특정 섹션 시작(주석임)

```
echo "Hello, world" # 특정 섹션 종료(주석임)
```

```
:leveloffset: 3
```

```
:leveloffset: +2
```

[ [autoinst-unattended] ]

= 무인 프로비저닝

"베어메탈" 기능을 사용하면 기본 PXE 부팅 이미지를 사용하여 새 시스템이 로컬 네트워크에 연결되자마자 해당 시스템을 등록할 수 있습니다. {productname} {webui}로 이동하여 이 시스템에 프로파일을 할당합니다. 다음에 클라이언트가 부팅되면 해당 프로파일에 따라 운영 체제가 설치됩니다. 베어메탈 프로비저닝에 대한 설명은 [xref:client-configuration:autoinst-unattended.adoc#bare-metal\[베어메탈 프로비저닝\]](#)을 참조하십시오.

베어메탈 기능을 사용하지 않으려면 {productname}에서 시스템을 수동으로 선언할 수 있습니다. {productname} API를 사용하면 베어메탈 기능을 사용하여 수집한 것처럼 시스템에 대한 시스템 레코드를 생성할 수 있습니다. API를 사용한 시스템 선언에 대한 설명은 [xref:client-configuration:autoinst-unattended.adoc#create-system-record\[수동으로 시스템 레코드 생성\]](#)을 참조하십시오.

[ [bare-metal] ]

== 베어메탈 조달

베어메탈 프로비저닝 옵션을 활성화한 경우 {productname} 네트워크에 연결된 모든 클라이언트는 전원이 켜지면 바로 조직에 자동으로 추가됩니다. 이 작업이 완료되면 클라이언트가 종료되고 [guimenu]``시스템`` 목록에 설치할 준비가 된 상태로 표시됩니다.

. 절차: 베어메탈 기능 활성화

- . {productname} {webui}에서 menu:관리[관리자 구성 > 베어메탈 시스템]으로 이동합니다.
- . btn:[이 조직에 추가 활성화]를 클릭합니다.

전원이 켜진 새로운 베어메탈 클라이언트는 베어메탈 기능을 활성화한 관리자에 속한 조직에 추가됩니다. 이러한 클라이언트는 "부트스트랩" 유형이며 일반 클라이언트로 전환되려면 프로비저닝되어야 합니다.

새 클라이언트가 추가될 조직을 변경하려면 베어메탈 기능을 비활성화하고 새 조직의 관리자로 로그인한 후 기능을 다시 활성화해야 합니다. [guilabel]``マイグ레이션`` 탭을 사용하여 이미 등록된 시스템을 다른 조직으로 마이그레이션할 수 있습니다.

이러한 방식으로 등록된 클라이언트를 사용하여 시스템 설정 관리자(SSM)을 사용할 수 있습니다. 그러나 해당 클라이언트에 운영 체제가 아직 설치되지 않았기 때문에 모든 SSM 기능을 베어메탈 클라이언트에 사용할 수 있는 것은 아닙니다. 또한 이러한 방식으로 등록된 시스템을 포함하는 혼합 세트에서도 마찬가지입니다. 세트의 모든 클라이언트가 프로비저닝되면 세트에 모든 기능을 사용할 수 있게 됩니다. SSM에 대한 자세한 설명은 [xref:client-configuration:system-set-manager.adoc\[\]](#)을 참조하십시오.

- . 절차: "부트스트랩" 유형 클라이언트 프로비저닝
- . {productname} {webui}에서 [guimenu]``시스템``으로 이동하여 프로비저닝할 클라이언트를 선택한 후 menu:프로비저닝[자동 설치] 탭으로 이동합니다.
- . 사용할 {ay} 프로파일을 선택하고 btn:[PXE 설치 구성 생성]을 클릭합니다. 이 옵션을 선택하면 Cobbler에서 시스템 항목이 생성됩니다.
- . 클라이언트의 전원을 켭니다.

서버는 TFTP를 사용하여 새 클라이언트를 프로비저닝하므로 프로비저닝에 성공하려면 적절한 포트 및 네트워크를 올바르게 구성해야 합니다.

**[[create-system-record]]**

= 수동으로 시스템 레코드 생성

API 호출을 사용하여 MAC 주소에 의해 식별되는 클라이언트와 자동 설치 프로파일 사이의 관계를 선언할 수 있습니다. 다음에 시스템이 재부팅되면 지정된 프로파일에 따라 설치가 시작됩니다.

- . 절차: 수동으로 선언된 프로파일에서 다시 설치

- . {productname} 서버의 명령 프롬프트에서

[systemitem]``system.createSystemRecord`` API를 호출합니다. 이 예에서 [literal]``name``을 클라이언트의 이름으로 바꾸고, [literal]``<profile>``을 프로파일 레이블로 바꾸며, [literal]``<iface>``를 [literal]``eth0``과 같이 클라이언트의 인터페이스 이름으로 바꾸고, [literal]``<hw\_addr>``을 [literal]``00:25:22:71:e7:c6``과 같은 하드웨어 주소로 바꿉니다.

+

```
$ spacecmd api --args '["<name>", "<profile>", "", "", \[{"name": "<iface>", "mac": "<hw_addr>"} \]]' \
system.createSystemRecord
```

- . 클라이언트의 전원을 켭니다. 네트워크에서 부팅되며 설치에 맞는 올바른 프로파일이 선택됩니다.

이 명령을 통해 Cobbler에서 시스템 레코드가 생성됩니다. 커널 옵션, 클라이언트의 IP 주소, 도메인 이름과 같은 추가 파라미터도 지정할 수 있습니다. 자세한 설명은 API 설명서에서 [systemitem]``createSystemRecord`` 호출``을 참조하십시오.

```
:leveloffset: 3
:leveloffset: +2
```

**[[autoinst-ownpgpkey]]**

= 자체 GPG 키 사용

자동 설치에서 사용할 리포지토리에 서명되지 않은 메타데이터가 있는 경우 일반적으로 [option]``insecure=1`` 커널 파라미터를 자동 설치 가능한 배포판 옵션으로 사용하고 {ay} 설치 파일에서 [path]``spacewalk/sles\_no\_signature\_checks`` 코드 조각을 사용해야 합니다.

더 안전한 방법은 자체 GPG 키를 제공하는 것입니다.

#### [NOTE]

=====

이 방법은 {suse} 클라이언트에만 해당합니다.

=====

- . 절차: 자체 GPG 키 포함
- . GPG 키를 생성합니다.
- . 이 키를 사용하여 패키지의 메타데이터에 서명합니다.
- . 설치 미디어의 초기 RAM 디스크에 추가합니다.

\* 키를 생성한 후 이 키를 사용하여 메타데이터에 서명하는 방법은 [xref:administration:repo-metadata.adoc](#)[리포지토리 메타데이터 서명]을 참조하십시오.  
 \* 네트워크 부팅을 위해 사용되는 설치 미디어에 키를 추가하는 방법은 [xref:client-configuration:autoinst-ownpgkey.adoc#gpg-key-pxeboot](#)[PXE 부팅을 위한 자체 GPG 키]를 참조하십시오.  
 \* CD-ROM에서 부팅하기 위해 사용되는 설치 미디어에 키를 추가하는 방법은 [xref:client-configuration:autoinst-ownpgkey.adoc#gpg-key- cdrom](#)[CD-ROM에서 자체 GPG 키]를 참조하십시오.

#### [NOTE]

=====

새 GPG 키를 사용하여 메타데이터에 서명한 후에는 이미 온보딩된 클라이언트는 새 키에 대해 알 수 없습니다. 이상적으로는 메타데이터에 서명한 후 클라이언트를 등록해야 합니다.

그러한 리포지토리를 사용하는 클라이언트를 이미 온보딩한 경우 해결 방법은 해당 클라이언트에서 GPG 키 확인을 비활성화하는 것입니다.

=====

#### [[gpg-key-pxeboot]]

-- PXE 부팅을 위한 자체 GPG 키

PXE 부팅 프로세스에서 사용되는 초기 RAM 디스크([path]``initrd``)에는 일반적으로 {suse}의 GPG 키만 포함됩니다. 이 파일에 자체 키를 추가해야 하며, 이 키는 패키지를 확인하는 데 사용됩니다.

- . 절차: 초기 RAM 디스크에 GPG 키 추가

- . GPG 키를 찾기 위해 부팅 프로세스 중 사용된 동일한 경로로 디렉토리를 생성합니다.
- +

```
mkdir -p tftpboot/usr/lib/rpm/gnupg/keys
```

- . 뒤에 [path]``.asc``를 추가하여 이 디렉토리에 GPG 키를 복사합니다.
- +

```
cp /srv/www/htdocs/pub/mgr-gpg-pub.key tftpboot/usr/lib/rpm/gnupg/keys/mgr-gpg-pub.asc
```

- . 최상위 디렉토리 내에서 컨텐트를 패키지화하고 설치 미디어 파일의 일부인 [path]``initrd``에 추가합니다.
- +

```
cd tftpboot find . | cpio -o -H newc | xz --check=crc32 -c >> /path/to/initrd
```

[[gpg-key-cdrom]]  
== CD-ROM에서 자체 GPG 키

[command]``mksusecd`` 유ти리티를 사용하여 설치 이미지를 수정할 수 있습니다. 이 유ти리티는 Development Tools 모듈에 포함되어 있습니다.

- . 절차: 설치 ISO 이미지에 GPG 키 추가
- . GPG 키를 찾기 위해 부팅 프로세스 중 사용된 동일한 경로로 디렉토리를 생성합니다.
- +

```
mkdir -p initrdroot/usr/lib/rpm/gnupg/keys
```

- . 뒤에 [path]``.asc``를 추가하여 이 디렉토리에 GPG 키를 복사합니다.
- +

```
cp /srv/www/htdocs/pub/mgr-gpg-pub.key initrdroot/usr/lib/rpm/gnupg/keys/mgr-gpg-pub.asc
```

- . [command]``mksusecd``를 사용하여 기존 ISO 이미지를 추가합니다.
- +

```
mksusecd --create <new-image>.iso --initrd initrdroot/ <old-image>.iso
```

```
:leveloffset: 3
:leveloffset: +1

[[virtualization]]
= 시각화
```

{productname}를 사용해 정규 기준 또는 Salt 클라이언트 외에 가상화 클라이언트도 관리할 수 있습니다. 이 설치 유형에서는 가상 호스트가 {productname} 서버에 설치되어 가상 게스트가 몇 개이든 모두 관리할 수 있습니다. 원하는 경우 가상 호스트를 몇 개 설치하여 게스트 그룹을 관리할 수 있습니다.

가상화 클라이언트가 보유한 기능의 범위는 사용자가 선택하는 타사 가상화 공급자에 따라 달라집니다.

Xen 및 KVM 호스트 및 게스트는 {productname}에서 직접 관리할 수 있습니다. 이를 통해 {ay} 또는 {kickstart}를 사용해 호스트 및 게스트를 자동 설치하고 {webui}에서 게스트를 관리할 수 있습니다.

VMWare vSphere를 포함한 VMWare와 Nutanix AHV의 경우 {productname}은 가상 호스트 관리자(VHM)를 설정해 VM을 제어할 것을 사용자에게 요구합니다. 이를 통해 사용자는 호스트 및 게스트에 대한 제어권을 얻게 됩니다. 하지만 Xen 및 KVM에서보다는 더 제한적입니다.

{productname}은(는) VMWare vSphere 또는 Nutanix AHV에서 VM을 생성하거나 편집할 수 없습니다.

```
//So I looked it up in their docs: "VMWare vSphere is a suite of
virtualization applications that includes ESXi and vCenter Server". So I
think using "VMWare vSphere" implies ESXi and vCenter without having to
spell them out. Happy to be proven wrong. --LKB 2019-07-10
```

다른 타사 가상화 공급자는 {productname}가 직접 지원하지 않습니다. 하지만 공급자가 VM에 대한 JSON 구성 파일을 엑스포트하도록 허용하는 경우 이 구성 파일을 {productname}로 업로드하여 VHM으로 관리할 수 있습니다.

VHM을 사용해 가상화를 관리하는 방법에 대한 자세한 내용은 [xref:client-configuration:vhm.adoc\[ \]](#)을 참조하십시오.

## == 가상화 호스트 관리

시작하기 전에 가상화 호스트로 사용하려는 클라이언트에 ``가상화 호스트`` 시스템 유형이 할당되었는지 확인하십시오. 기존 클라이언트와 Salt 클라이언트 모두 가상 호스트로 사용할 수 있습니다. `menu:시스템[시스템 목록]`으로 이동하여 가상화 호스트로 사용할 클라이언트의 이름을 클릭합니다. 시스템 유형은 [guimenu]``시스템 등록 정보`` 섹션에 나열되어 있습니다. ``가상화 호스트`` 시스템 유형이 나열되어 있지 않은 경우 `btn:[이 등록 정보 편집]`을 클릭하여 이 유형을 할당하십시오.

클라이언트에 ``가상화 호스트`` 시스템 유형이 있는 경우 클라이언트의 시스템 정보 페이지에서 [guimenu] ``가상화`` 탭을 제공합니다. [guimenu] ``가상화`` 탭을 통해 가상 게스트를 생성 및 관리하고 저장소 풀 및 가상 네트워크를 관리할 수 있습니다.

#### == 가상 게스트 생성

{productname} {webui}에서 가상화 호스트에 가상 게스트를 추가할 수 있습니다.

##### . 절차: 가상 게스트 생성

- . {productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 가상화 호스트의 이름을 클릭하고, [guimenu] ``가상화`` 탭으로 이동합니다.
- . [guimenu] ``일반`` 섹션에서 다음과 같은 상세 정보를 입력합니다.
  - + \* [guimenu] ``게스트`` 하위 탭에서 btn:[게스트 생성]을 클릭합니다.
  - \* [guimenu] ``이름`` 필드에서 게스트의 이름을 입력합니다.
  - \* [guimenu] ``하이퍼바이저`` 필드에서 사용할 하이퍼바이저를 선택합니다.
  - \* [guimenu] ``가상 머신 유형`` 필드에서 완전 가상화 또는 반가상화를 선택합니다.
  - \* [guimenu] ``최대 메모리`` 필드에 게스트 디스크 크기의 상한을 MiB 단위로 입력합니다.
  - \* [guimenu] ``가상 CPU 수``에서 게스트를 위한 vCPU의 수를 입력합니다.
  - \* [guimenu] ``아키텍처`` 필드에서 게스트에서 사용할 에뮬레이트된 CPU 아키텍처를 선택합니다. 선택한 아키텍처는 기본적으로 가상 호스트와 일치합니다.
  - \* [guimenu] ``자동 설치 프로파일`` 필드에서 게스트를 설치하는 데 사용할 자동 설치 도구를 선택합니다. 자동 설치를 사용하지 않으려면 이 필드를 공백으로 두십시오.
  - . [guimenu] ``디스크`` 섹션에서 클라이언트와 함께 사용할 가상 디스크의 상세 정보를 입력합니다. [guimenu] ``소스 템플릿 이미지 URL`` 필드에 운영 체제 이미지의 경로를 반드시 입력해야 합니다. 입력하지 않으면 게스트가 빈 디스크로 생성됩니다.
  - . [guimenu] ``네트워크`` 섹션에서 클라이언트와 함께 사용할 가상 네트워크 인터페이스의 상세 정보를 입력합니다. MAC 주소를 생성하려면 [guimenu] ``MAC 주소`` 필드를 공백으로 두십시오.
  - . [guimenu] ``그래픽`` 섹션에서 클라이언트와 함께 사용할 그래픽 드라이버의 상세 정보를 입력합니다.
  - . 생성할 게스트의 일정을 잡고 btn:[생성]을 클릭하여 게스트를 생성합니다.
  - . 새로운 가상 게스트는 생성되자마자 시작합니다.

{productname} {webui} 내에서 pacemaker 클러스터에도 가상 게스트를 추가할 수 있습니다.

##### . 절차: 클러스터 관리형 가상 게스트 생성

- . 다음을 추가하여 클러스터 노드 중 하나에서 ``가상 게스트 만들기`` 절차를 따르십시오.
  - + \* [guimenu] ``클러스터 리소스로 정의`` 필드가 선택되어 있는지 확인합니다.

- \* [guimenu] ``VM 정의를 위한 클러스터 공유 폴더 경로`` 필드에 게스트 구성이 저장될 모든 클러스터 노드가 공유하는 폴더의 경로를 입력합니다.
- \* 모든 클러스터 노드에서 공유하는 저장소 폴에 모든 디스크가 있는지 확인합니다.

클러스터에서 관리하는 가상 게스트는 라이브 마이그레이션이 가능합니다.

```
:leveloffset: 3
:leveloffset: +2
```

[ [virt-xenkvm]]  
= Xen 및 KVM을 이용한 가상화

Xen 및 KVM 가상화 클라이언트는 {productname}에서 직접 관리할 수 있습니다.

먼저 {productname} 서버에서 가상 호스트를 설정해야 합니다. 그런 다음, 향후 가상 호스트 및 가상 게스트에 대해 {ay} 또는 {kickstart}를 사용해 자동 설치를 설정할 수 있습니다.

이 섹션에는 가상 게스트 설치 후 이 게스트를 관리하는 방법에 대한 정보도 포함되어 있습니다.

#### == 호스트 설정

VM 호스트에서 Xen 또는 KVM을 설정하는 방법은 연결된 게스트에서 사용하려는 운영 체제에 따라 달라집니다.

{suse} 운영 체제의 경우 <https://documentation.suse.com/sles/15-SP3/html/SLES-all/book-virtualization.html>[]에서 사용 가능한 SLES 가상화 안내서를 참조하십시오.

{rhel} 운영 체제의 경우 해당 버전의 Red Hat 문서를 참조하십시오.

{productname}은 [systemitem] ``libvirt``를 사용해 게스트를 설치하고 관리합니다. 호스트에 [daemon] ``libvirdt`` 패키지 설치를 완료한 상태이어야 합니다. 대부분의 경우 대개 기본 설정으로 충분하므로 기본 설정을 조정하지 않아도 됩니다. 하지만 게스트에서 루트가 아닌 사용자로 VNC 콘솔에 액세스하려면 구성을 몇 가지 변경해야 합니다. 이를 설정하는 방법에 대한 자세한 내용은 해당 운영 체제의 관련 문서를 참조하십시오.

{productname} 서버에서 부트스트랩 스크립트가 필요합니다. 부트스트랩 스크립트는 호스트를

위한 활성화 키를 포함해야 합니다. 또한 추가 보안을 위해 GPG 키를 포함하는 것이 좋습니다. 부트스트랩 스크립트 생성에 대한 자세한 내용은 [xref:client-configuration:registration-bootstrap.adoc\[ \]](#)을 참조하십시오.

부트스트랩 스크립트가 준비되었으면 호스트에서 실행하여 `{productname}` 서버로 등록합니다. 클라이언트 등록에 대한 자세한 내용은 [xref:client-configuration:registration-overview.adoc\[ \]](#)을 참조하십시오.

Salt 클라이언트의 경우 `[systemitem]``가상화 호스트``` 자격을 활성화해야 합니다. 이렇게 하면 VM 변경 사항을 즉시 볼 수 있습니다. 이를 위해서는 `{productname}` `{webui}`에서 호스트의 `[guimenu]``시스템 정보``` 페이지로 이동하여 `[guimenu]``등록 정보``` 탭을 클릭하십시오. 또는 등록 키 수준에서 `[systemitem]``가상화 호스트``` 자격을 추가할 수 있습니다. `[guimenu]``추가 시스템 유형``` 섹션에서 `[guimenu]``가상화 호스트```의 확인란을 선택하고 `btn:[등록 정보 업데이트]`를 클릭하여 변경 사항을 저장합니다. 다음과 같이 Salt Minion 서비스를 다시 시작하여 변경 사항을 활성화합니다.

```
systemctl restart salt-minion
```

기존 클라이언트의 경우 VM 호스트는 기본적으로 `[systemitem]``rhnsd``` 서비스를 사용해 예약된 작업을 확인합니다. 이러한 확인은 네 시간마다 이루어지며, 이를 통해 다수의 클라이언트가 있는 환경에서 로드를 밸런싱합니다. 이로 인해 작업이 수행될 때까지 최대 네 시간의 지연이 발생할 수 있습니다. VM 게스트를 관리하고 있다면 이처럼 긴 지연 시간이 항상 좋은 것은 아닙니다(특히 게스트 재부팅과 같은 작업일 경우). 이 문제를 해결하려면 `[systemitem]``rhnsd``` 서비스를 비활성화하고 `[daemon]``osad``` 서비스를 활성화하면 됩니다. `[daemon]``osad``` 서비스는 jabber 프로토콜을 사용해 명령을 수신하고 명령을 즉시 실행합니다.

`[systemitem]``rhnsd``` 서비스를 비활성화하고 `[daemon]``osad``` 데몬을 활성화하려면 루트 사용자로 다음 명령을 실행하십시오.

```
service rhnsd stop service rhnsd disable
```

```
service osad enable service osad start
```

== 자동 설치

`{ay}` 또는 `{kickstart}`를 사용해 Xen 및 KVM 게스트를 자동으로 설치하고 등록할 수 있습니다.

게스트를 등록하려는 VM 호스트와 각 게스트에 활성화 키가 필요합니다. 활성화 키에는 `[systemitem]``조달``` 및 `[systemitem]``가상화 플랫폼``` 자격이 있어야 합니다. 활성화

키에는 [package]``mgr-virtualization-host`` 및 [package]``mgr-osad`` 패키지에 대한 액세스 권한도 있어야 합니다. 활성화 키 생성에 대한 자세한 내용은 [xref:client-configuration:activation-keys.adoc\[ \]](#)를 참조하십시오.

설치 후 {productname}로 게스트를 자동 등록하려면 부트스트랩 스크립트를 생성해야 합니다. 부트스트랩 스크립트 생성에 대한 자세한 내용은 [xref:client-configuration:registration-bootstrap.adoc\[ \]](#)을 참조하십시오.

[IMPORTANT]

=====

VM 게스트 자동 설치는 기존 클라이언트로 구성된 경우에만 작동합니다. Salt 클라이언트는 {ay} 또는 {kickstart}가 아닌 템플릿 디스크 이미지를 사용해 생성할 수 있습니다.

=====

==== 자동 설치 가능한 배포판 생성

{productname}에서 클라이언트를 자동 설치하려면 VM 호스트에 자동 설치 가능한 배포를 생성해야 합니다. 마운트한 로컬 또는 원격 디렉토리나 루프 마운트 ISO 이미지에서 배포를 사용할 수 있는 상태로 만들 수 있습니다.

자동 설치 가능한 배포의 구성은 게스트에서 SLES 또는 {rhe1} 운영 체제를 사용하고 있는지 여부에 따라 달라집니다. {rhe1} 설치를 위한 패키지는 연결된 기본 채널에서 가져옵니다. {suse} 시스템 설치를 위한 패키지는 자동 설치 가능한 배포에서 가져옵니다. 따라서 SLES 시스템의 경우 자동 설치 가능한 배포는 완전한 설치 소스이어야 합니다.

.자동 설치 가능한 배포를 위한 경로

[cols="1,1,1", options="header"]

|====

|                                            |
|--------------------------------------------|
| 운영 체제 유형   커널 위치   initrd 위치               |
| {rhe1}   [path]``images/pxeboot/vmlinuz``  |
| [path]``images/pxeboot/initrd.img``        |
| SLES   [path]``boot/<arch>/loader/initrd`` |
| [path]``boot/<arch>/loader/linux``         |
| ====                                       |

모든 경우 기본 채널이 자동 설치 배포와 일치하는지 확인하십시오.

시작하기 전에 VM 호스트에 사용할 수 있는 설치 미디어가 있는지 확인하십시오. 이러한 설치 미디어는 네트워크 리소스, 로컬 디렉토리 또는 루프 마운트 ISO 이미지에 있을 수 있습니다. 또한 모든 파일 및 디렉토리가 누구나 읽을 수 있는 것인지 확인하십시오.

.절차: 자동 설치 가능한 배포 생성

- . {productname} {webui}에서 menu:시스템[자동 설치 > 배포판]으로 이동한 후 btn:[배포]

생성]을 클릭합니다.

. [guimenu]``자동 설치 가능한 배포판 생성`` 섹션에서 다음 파라미터를 사용합니다.

\* [guimenu]``배포판 레이블`` 섹션에 배포판의 고유 이름을 입력합니다.

글자, 숫자, 하이픈(``-``), 마침표(``.`), 밑줄(``\_``)만 사용하고, 이름을 구성하는 문자는 다섯 개 이상이어야 합니다.

\* [guimenu]``트리 경로`` 필드에 설치 원본의 절대 경로를 입력합니다.

\* [guimenu]``기본 채널`` 필드에서 설치 소스와 일치하는 채널을 선택합니다.

이 채널은 비{suse} 설치에 대해 패키지 소스로 사용됩니다.

\* [guimenu]``설치 프로그램 생성`` 필드에서 설치 소스와 일치하는 운영 체제 버전을 선택합니다.

\* [guimenu]``커널 옵션`` 필드에는, 설치를 위해 부팅할 때 커널로 전달될 옵션을 입력합니다.

[option]``install`` 파라미터와 [option]``self\_update=0`` pt.options=self\_update`` 파라미터가 기본적으로 추가됩니다.

\* [guimenu]``커널 후 옵션`` 섹션에는, 설치된 시스템을 처음 부팅할 때 커널로 전달될 옵션을 입력합니다.

. btn:[자동 설치 가능한 배포판 생성]을 클릭하여 저장합니다.

자동 설치 가능한 배포판을 생성했으면 menu:시스템[자동 설치 > 배포판]으로 이동하여 편집할 배포판을 선택하고 편집할 수 있습니다.

#### ==== 자동 설치 프로파일 생성 및 업로드

자동 설치 프로파일에는 시스템을 설치하는 데 필요한 모든 설치 및 구성 데이터가 포함되어 있습니다. 설치 완료 후 실행될 스크립트도 포함할 수 있습니다.

{kickstart} 프로파일은 {productname} {webui}에서 menu:시스템[자동 설치 > 프로파일]로 이동하여 btn:[새 Kickstart 파일 생성]을 클릭하고 표시되는 프롬프트에 따라 생성할 수 있습니다.

{ay} 또는 {kickstart} 자동 설치 프로파일을 수동으로 생성할 수도 있습니다. {suse}는 사용자 정의 파일의 시작점으로 사용할 수 있는 {ay} 설치 파일의 템플릿을 제공합니다.

템플릿은 link:<https://github.com/SUSE/manager-build-profiles>[ ]에서 확인할 수 있습니다.

{ay}를 사용하여 SLES를 설치하는 경우 다음 코드 조각도 포함해야 합니다.

```
<products config:type=\list> <listentry>SLES</listentry> </products>
```

\* {ay}에 대한 설명은 xref:client-configuration:autoinst-profiles.adoc#autoyast[ ]를 참조하십시오.

\* {kickstart}에 대한 자세한 설명은 xref:client-configuration:autoinst-profiles.adoc#kickstart[ ]를 참조하거나 해당 설치에 대한 Red Hat 문서를 참조하십시오.

### . 절차: 자동 설치 프로파일 업로드

- . {productname} {webui}에서 menu:시스템[자동 설치 > 프로파일]로 이동하여 btn:[Kickstart/AutoYaST 파일 업로드]를 클릭합니다.
- . [guimenu]``자동 설치 프로파일 생성`` 섹션에서 다음 파라미터를 사용합니다.
- \* [guimenu]``레이블`` 필드에 프로파일의 고유 이름을 입력합니다.  
글자, 숫자, 하이픈(``-``), 마침표(``.`), 밑줄(``\_``)만 사용하고, 이름을 구성하는 문자는 일곱 개 이상이어야 합니다.
- \* [guimenu]``자동 설치 트리`` 필드에서 앞서 생성한 자동 설치 가능한 배포를 선택합니다.
- \* [guimenu]``가상화 유형`` 필드에서 해당되는 게스트 유형(예: [parameter]``KVM 가상화 게스트``)을 선택합니다.  
여기에서 [guimenu]``Xen 가상화 호스트``를 선택하지 마십시오.
- \* 옵션: 자동 설치 프로파일을 수동으로 생성하려면 [guimenu]``파일 내용`` 필드에 프로파일을 직접 입력할 수 있습니다.  
파일을 이미 생성한 경우 [guimenu]``파일 내용`` 필드를 공백으로 두십시오.
- \* [guimenu]``업로드할 파일`` 필드에서 btn:[파일 선택]을 클릭하고 시스템 대화 상자를 사용해 업로드할 파일을 선택합니다.  
파일 업로드가 완료되면 파일 이름이 [guimenu]``업로드할 파일`` 필드에 표시됩니다.
- \* 업로드한 파일의 내용이 [guimenu]``파일 내용`` 필드에 표시됩니다.  
편집하려면 직접할 수 있습니다.
- . btn:[생성]을 클릭하여 변경 사항을 저장하고 프로파일을 보관합니다.

자동 설치 프로파일을 생성했으면 menu:시스템[자동 설치 > 프로파일]로 이동해 편집하려는 프로파일을 선택하여 편집할 수 있습니다. 원하는 대로 변경한 후 btn:[생성]을 클릭하여 설정을 저장합니다.

#### [IMPORTANT]

=====

기존 {kickstart} 프로파일의 [guimenu]``가상화 유형``을 변경하면 부트로더 및 파티션 옵션이 수정되어 사용자 정의 설정이 덮어쓰기될 수도 있습니다. [guimenu]``파티셔닝`` 탭을 주의 깊게 검토하여 이 설정을 확인한 후에 변경하십시오.

=====

#### ==== 게스트를 자동으로 등록

VM 게스트를 자동으로 설치하는 경우 {productname}에 등록되지 않습니다. 게스트가 설치되자마자 자동으로 등록되게 하려면 부트스트랩 스크립트를 호출하고 게스트를 등록하는 자동 설치 프로파일에 섹션을 추가할 수 있습니다.

이 섹션에서는 부트스트랩 스크립트를 기존 {ay} 프로파일에 추가하는 것에 관한 지침을 제공합니다.

부트스트랩 스크립트 생성에 대한 자세한 내용은 [xref:client-configuration:registration-bootstrap.adoc](#)[ ]을 참조하십시오. {kickstart}에 대해 이 작업을 하는 방법에 대한 지침은 해당 설치에 대한 Red Hat 문서를 참조하십시오.

. 절차: 부트스트랩 스크립트를 {ay} 프로파일에 추가

- . 등록하려는 VM 게스트에 대한 활성화 키를 부트스트랩 스크립트가 포함하는지, 이 키가 [path]``/srv/www/htdocs/pub/bootstrap\_vm\_guests.sh``의 호스트에 있는지 확인합니다.
- . {productname} {webui}에서 menu:시스템[자동 설치 > 프로파일]로 이동하여 이 스크립트를 연결할 {ay} 프로파일을 선택합니다.
- . [guimenu]``파일 내용`` 필드에서 이 코드 조각을 파일 끝의 종료 태그 ``</profile>`` 바로 앞에 추가합니다.

아래 코드 조각의 예시 IP 주소를 {productname} 서버의 정확한 IP 주소로 교체해야 합니다.

+  
+

```
<scripts> <init-scripts config:type="list"> <script> <interpreter>shell </interpreter> <location> http:// 192.168.1.1 /pub/bootstrap/bootstrap_vm_guests.sh </location> </script> </init-scripts> </scripts>
```

+  
+

- . menu:업데이트[ ]를 클릭하여 변경 사항을 저장합니다.

[IMPORTANT]

=====

{ay} 프로파일에 ``<scripts>`` 섹션이 이미 포함되어 있는 경우 이 섹션을 추가하지 마십시오. 기존 ``<scripts>`` 섹션 안에 부트스트랩 코드 조각을 배치합니다.

=====

==== VM 게스트 자동 설치

모든 것을 설정했으면 VM 게스트 자동 설치를 시작할 수 있습니다.

[IMPORTANT]

=====

각 VM 호스트는 한 번에 한 게스트만 설치할 수 있습니다. 두 건 이상의 자동 설치 일정을 잡는 경우 이전 설치가 완료되기 전에 다음번 설치가 시작되지 않도록 시간을 안배해야 합니다. 다른 게스트 자동 설치가 아직 실행 중일 때 게스트 설치가 시작되면 실행 중인 설치가 취소됩니다.

=====

- . {productname} {webui}에서 menu:시스템[개요]로 이동하여 게스트를 설치하려는 VM

호스트를 선택합니다.

- . [guiitem] ``가상화`` 탭의 [guimenu] ``조달`` 하위 탭으로 이동합니다.
- . 사용하려는 자동 설치 프로파일을 선택하고, 게스트에 고유한 이름을 지정합니다.
- . 해당되는 경우 프록시를 선택하고 일정을 입력합니다.
- . 게스트의 하드웨어 프로파일 및 구성 옵션을 변경하려면 btn:[고급 옵션]을 클릭합니다.
- . btn:[자동 설치 일정 잡기 후 완료]를 클릭하여 완료합니다.

== VM 게스트 관리

{productname} {webui}를 사용해 종료, 재시작, CPU 및 메모리 할당 조정 등의 작업을 포함해 VM 게스트를 관리할 수 있습니다.

이를 위해서는 {productname} 서버에 등록된 Xen 또는 KVM VM 호스트가 필요하며 호스트에 [daemon] ``libvirtd`` 서비스가 실행 중이어야 합니다. 기존 클라이언트의 경우 {productname} 서버에 설치된 [package] ``mgr-cfg-actions`` 패키지도 필요합니다.

{productname} {webui}에서 menu:시스템[시스템 목록]으로 이동하여 관리하려는 게스트의 VM 호스트를 클릭합니다. [guimenu] ``가상화`` 탭으로 이동하여 이 호스트에 등록된 모든 게스트를 확인하고 관리 기능에 액세스합니다.

{webui}를 사용한 VM 게스트 관리에 대한 자세한 내용은

[xref:reference:systems/system-details/sd-virtualization.adoc](#)[ ]를 참조하십시오.

:leveloffset: 3  
:leveloffset: +1

[[virt-vhm]]

= 가상 호스트 관리자

가상 호스트 관리자(VHM)는 다양한 클라이언트 유형에서 정보를 수집하는 데 사용됩니다.

VHM을 사용해 개인 또는 공용 클라우드 인스턴스를 수집하여 가상화 그룹으로 편성할 수 있습니다. 가상화 클라이언트를 이렇게 편성한 상태에서 Taskomatic은 {productname} {webui}에 표시할 클라이언트의 데이터를 수집합니다. VHM을 통해 가상화 클라이언트에서 일치하는 구독을 사용할 수도 있습니다.

{productname} 서버에서 VHM을 생성한 후 이를 사용해 사용 가능한 공용 클라우드 인스턴스를 목록에 포함할 수 있습니다. VHM을 사용해 {kube}로 생성한 클러스터를 관리할 수도 있습니다.

```
// We could probably use a diagram here, to convey the meaning behind
this:
// Virtual Host Managers (VHMs) can be used to manage one or more virtual
hosts.
// Virtual Hosts are hypervisors provided by a third party.
```

```
// Each virtual host can contain one or more virtual guests.
// --LKB 2017-07-15

* Amazon Web Services에서 VHM을 사용하는 방법에 대한 자세한 내용은 xref:client-
configuration:vhm-aws.adoc[]를 참조하십시오.
* Microsoft Azure에서 VHM을 사용하는 방법에 관한 자세한 내용은 xref:client-
configuration:vhm-azure.adoc[]를 참조하십시오.
* Google Compute Engine에서 VHM을 사용하는 방법에 대한 자세한 내용은 xref:client-
configuration:vhm-gce.adoc[]를 참조하십시오.
* {kube}에서 VHM을 사용하는 방법에 대한 자세한 내용은 xref:client-
configuration:vhm-kubernetes.adoc[]를 참조하십시오.
* Nutanix에서 VHM을 사용하는 방법에 대한 자세한 내용은 xref:client-
configuration:vhm-nutanix.adoc[]를 참조하십시오.
* VMWare vSphere에서 VHM을 사용하는 방법에 대한 자세한 내용은 xref:client-
configuration:vhm-vmware.adoc[]를 참조하십시오.
* 다른 호스트에서 VHM을 사용하는 방법에 대한 자세한 내용은 xref:client-
configuration:vhm-file.adoc[]를 참조하십시오.
```

:leveloffset: 3  
:leveloffset: +2

[[vhm-aws]]  
= VHM 및 Amazon Web Services

가상 호스트 관리자(VHM)을 사용하여 Amazon Web Services(AWS)에서 인스턴스를 수집할 수 있습니다.

VHM을 통해 {productname}는 클러스터에 대한 정보를 획득하고 보고할 수 있습니다. VHM에 대한 자세한 내용은 xref:client-configuration:vhm.adoc[ ]를 참조하십시오.

== Amazon EC2 VHM 생성

가상 호스트 관리자(VHM)는 {productname} 서버에서 실행됩니다.

{productname} 서버에 [systemitem]``virtual-host-gatherer-libcloud`` 패키지를 설치했는지 확인하십시오.

. 절차: Amazon EC2 VHM 생성

- . {productname} {webui}에서 menu:시스템[가상 호스트 관리자]로 이동합니다.
- . btn:[생성]을 클릭하고 드롭다운 메뉴에서 [guimenu]``Amazon EC2``를 선택합니다.
- . [guimenu]``Amazon EC2 가상 호스트 관리자 추가`` 섹션에서 다음과 같은 파라미터를

사용합니다.

- \* [guimenu] ``레이블`` 필드에 VHM의 사용자 정의 이름을 입력합니다.
- \* [guimenu] ``액세스 키 ID`` 필드에서 Amazon이 제공하는 액세스 키 ID를 입력합니다.
- \* [guimenu] ``비밀 액세스 키`` 필드에 Amazon 인스턴스에 연결된 비밀 액세스 키를 입력합니다.
- \* [guimenu] ``지역`` 필드에 사용할 지역을 입력합니다.
- \* [guimenu] ``영역`` 필드에 VM이 있는 영역을 입력합니다. 이 영역은 작업과 일치하는 구독에 필요합니다. 지역 및 영역 설정에 대한 자세한 설명은 [xref:client-configuration:virtualization.adoc#\\_susesupport\\_and\\_vm\\_zones\[ \]](#)를 참조하십시오.
- . btn:[생성]을 클릭하여 변경 사항을 저장하고 VHM을 생성합니다.
- . [guimenu] ``가상 호스트 관리자`` 페이지에서 새 VHM을 선택합니다.
- . [guimenu] ``등록 정보`` 페이지에서 btn:[데이터 새로 고침]을 클릭하여 새 VHM을 목록에 포함합니다.

어떤 객체 및 리소스가 목록에 포함되었는지 확인하려면 menu:시스템[시스템 목록 > 가상 시스템]으로 이동하십시오.

Amazon 공용 클라우드에서 실행되는 인스턴스는 ``i`` 뒤에 17개의 16진 숫자가 나오는 형식으로 {productname} 서버에 UUID를 보고합니다.

I1234567890abcdef0

#### == 가상 호스트 관리자에 대한 AWS 권한

보안상의 이유로 항상 수행할 작업에 가능한 최소 권한을 부여해야 합니다. 사용자에게 과도한 권한이 있는 액세스 키를 사용하여 AWS에 연결하는 것은 권장되지 않습니다.

SUSE Manager가 AWS에서 필요한 정보를 수집하기 위해서는 VHM에 EC2 인스턴스 및 주소를 설명할 수 있는 권한이 필요합니다. 이러한 권한을 부여할 수 있는 한 가지 방법은 이 작업과 관련된 새 IAM 사용자(ID 및 액세스 관리)를 생성하고 다음과 같이 정책을 생성한 후 사용자에게 연결하는 것입니다.

```
{ "버전": "2012-10-17", "설명": [{ "효과": "허용", "동작": ["ec2:DescribeAddresses", "ec2:DescribeInstances"], "리소스": "*" }] }
```

특정 영역에 대한 액세스를 제한하여 권한을 자세하게 제한할 수 있습니다. 자세한 내용은 [https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ExamplePolicies\\_EC2.html#iam-example-read-only\[ \]](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ExamplePolicies_EC2.html#iam-example-read-only[ ])를 참조하십시오.

```
:leveloffset: 3
:leveloffset: +2
```

```
[[vhm-azure]]
```

## = VHM 및 Azure

가상 호스트 관리자(VHM)을 사용하여 Microsoft Azure에서 인스턴스를 수집할 수 있습니다.

VHM을 통해 {productname}는 가상 머신에 대한 정보를 획득하고 보고할 수 있습니다. VHM에 대한 자세한 내용은 [xref:client-configuration:vhm.adoc](#)[ ]을 참조하십시오.

### == 전제 조건

사용자가 생성하는 VHM은 Azure VM에 액세스하려면 올바른 권한이 할당되어 있어야 합니다.

Azure 계정에 구독 관리자로 로그인하여 Azure 사용자 계정 및 응용 프로그램이 올바른 그룹에 있는지 확인하십시오. 응용 프로그램이 속한 그룹에 따라 응용 프로그램의 역할이 결정되고, 이에 따라 권한도 결정됩니다.

### == Azure VHM 생성

가상 호스트 관리자(VHM)는 {productname} 서버에서 실행됩니다.

{productname} 서버에 [systemitem]``virtual-host-gatherer-libcloud`` 패키지를 설치했는지 확인하십시오.

#### . 절차: Azure VHM 생성

- . {productname} {webui}에서 menu:시스템[가상 호스트 관리자]로 이동합니다.
- . **btn:[생성]**을 클릭하고 드롭다운 메뉴에서 [guimenu]``Azure``를 선택합니다.
- . [guimenu]``Azure 가상 호스트 관리자 추가`` 섹션에서 다음과 같은 파라미터를 사용합니다.
  - \* [guimenu]``레이블`` 필드에 VHM의 사용자 정의 이름을 입력합니다.
  - \* [guimenu]``구독 ID`` 필드에서 [path]``Azure 포털 > 서비스 > 구독`` 페이지에서 찾은 구독 ID를 입력합니다.
  - \* [guimenu]``응용 프로그램 ID`` 필드에 응용 프로그램 등록 시 수집한 응용 프로그램 ID를 입력합니다.
  - \* [guimenu]``테넌트 ID`` 필드에서 응용 프로그램을 등록할 때 수집한 Azure에서 제공한 테넌트 ID를 입력합니다.
  - \* [guimenu]``비밀 키`` 필드에 Azure 인스턴스에 연결된 비밀 키를 입력합니다.
  - \* [guimenu]``영역`` 필드에 VM이 있는 영역을 입력합니다. 예를 들어, 서유럽의 경우 [path]``westeurope``을 입력합니다.
- 이 키는 작업과 일치하는 등록을 위해 필요합니다.
- . **btn:[생성]**을 클릭하여 변경 사항을 저장하고 VHM을 생성합니다.
- . [guimenu]``가상 호스트 관리자`` 페이지에서 새 VHM을 선택합니다.
- . [guimenu]``등록 정보`` 페이지에서 **btn:[데이터 새로 고침]**을 클릭하여 새 VHM을 목록에 포함합니다.

어떤 객체 및 리소스가 목록에 포함되었는지 확인하려면 menu:시스템[시스템 목록 > 가상 시스템]으로 이동하십시오.

== 권한 할당

// OM 2022-02-28: if the UI suggestion has been implemented (#1170298 via

#1170514) we may eventually be able to remove this section

권한이 올바르게 설정되지 않으면 [command]``virtual-host-gatherer``를 실행할 때 다음과 같은 오류를 수신할 수 있습니다.

일반 오류: [AuthorizationFailed] 객체 ID가 'object\_ID' 인 'client\_name' 클라이언트에 '/subscriptions/not-very-secret-subscription-id' 범위에 걸쳐 'Microsoft.Compute/virtualMachines/read' 작업을 수행할 권한이 없거나 이 범위가 잘못되었습니다. 액세스가 최근에 허용되었다면 인증서를 새로 고침하시기 바랍니다.

인증서가 올바른지 확인하려면 다음과 같이 {productname} 서버의 프롬프트에서 다음 명령을 실행하십시오.

```
virtual-host-gatherer -i input_azure.json -o out_azure.json -vvv
```

[path]``input\_azure.json`` 파일은 다음과 같은 정보를 포함해야 합니다.

```
[{ "id": "azure_vhm", "module": "Azure", "subscription_id": "subscription-id", "application_id": "application-id", "tenant_id": "tenant-id", "secret_key": "secret-key", "zone": "zone" }]
```

== Azure UUID

Azure 공용 클라우드에서 실행되는 인스턴스는 다음과 같이 이 UUID를 {productname} 서버에 보고합니다.

13f56399-bd52-4150-9748-7190aae1ff21

```
:leveloffset: 3
:leveloffset: +2
```

[ [vhm-hce]]  
= VHM 및 Google Compute Engine

가상 호스트 관리자(VHM)을 사용하여 Google Compute Engine(GCE)에서 인스턴스를 수집할 수 있습니다.

VHM을 통해 {productname}는 가상 머신에 대한 정보를 획득하고 보고할 수 있습니다. VHM에 대한 자세한 내용은 [xref:client-configuration:vhm.adoc](#)[ ]을 참조하십시오.

== 전제 조건

사용자가 생성하는 VHM은 GCE VM에 액세스하려면 올바른 권한이 할당되어 있어야 합니다.

Google 클라우드 플랫폼 계정에 관리자로 로그인하여 Cloud Identity and Access Management(IAM) 도구를 사용해 서비스 계정에 적절한 역할이 있는지 확인합니다.

#### == GCE VHM 생성

가상 호스트 관리자(VHM)는 {productname} 서버에서 실행됩니다.

VHM을 실행하려면 {productname} 서버가 포트 443을 열어 클라이언트에 액세스해야 합니다.

{productname} 서버에 [systemitem]``virtual-host-gatherer-libcloud`` 패키지를 설치했는지 확인하십시오.

시작하기 전에 GCE 패널에 로그인하여 인증서 파일을 다운로드합니다. 이 파일을 {productname} 서버에 로컬로 저장하고 경로를 적어둡니다.

#### . 절차: GCE VHM 생성

- . {productname} {webui}에서 menu:시스템[가상 호스트 관리자]로 이동합니다.
- . btn:[생성]을 클릭하고 드롭다운 메뉴에서 [guimenu]``Google Compute Engine``을 선택합니다.
- . [guimenu]``Google Compute Engine 가상 호스트 관리자 추가`` 섹션에서 다음과 같은 파라미터를 사용합니다.
  - \* [guimenu]``레이블`` 필드에 VHM의 사용자 정의 이름을 입력합니다.
  - \* [guimenu]``서비스 계정 전자 메일`` 필드에 서비스 계정에 연결된 이메일 주소를 입력합니다.
  - \* [guimenu]``인증서 경로`` 필드에 {productname} 서버의 로컬 경로를 GCE 패널에서 다운로드한 키에 입력합니다.
  - \* [guimenu]``프로젝트 ID`` 필드에 GCE 인스턴스가 사용하는 프로젝트 ID를 입력합니다.
  - \* [guimenu]``영역`` 필드에 VM이 있는 영역을 입력합니다.
 이 키는 작업과 일치하는 등록을 위해 필요합니다.
- . btn:[생성]을 클릭하여 변경 사항을 저장하고 VHM을 생성합니다.
- . [guimenu]``가상 호스트 관리자`` 페이지에서 새 VHM을 선택합니다.
- . [guimenu]``등록 정보`` 페이지에서 btn:[데이터 새로 고침]을 클릭하여 새 VHM을 목록에 포함합니다.

어떤 객체 및 리소스가 목록에 포함되었는지 확인하려면 menu:시스템[시스템 목록 > 가상 시스템]으로 이동하십시오.

#### == 권한 할당

권한이 올바르게 설정되지 않으면 [command]``virtual-host-gatherer``를 실행할 때 다음과 같은 오류를 수신할 수 있습니다.

오류: {'domain': 'global', 'reason': 'forbidden', 'message': "Required 'compute.zones.list' permission for"}

'projects/project-id"} 오류: 지정된 인증서를 사용해 Google Compute Engine 공용 클라우드에 연결할 수 없습니다.

인증서가 올바른지 확인하려면 다음과 같이 {productname} 서버의 프롬프트에서 다음 명령을 실행하십시오 .

```
virtual-host-gatherer -i input_google.json -o out_google.json -vvv
```

[path]``input\_google.json`` 파일은 다음과 같은 정보를 포함해야 합니다.

```
[{ "id": "google_vhm", "module": "GoogleCE", "service_account_email": "mail@example.com", "cert_path": "secret-key", "project_id": "project-id", "zone": "zone" }]
```

== GCE UUID

Google 공용 클라우드에서 실행되는 인스턴스는 다음과 같이 이 UUID를 {productname} 서버에 보고합니다.

152986662232938449

```
:leveloffset: 3
:leveloffset: +2
```

[ [kubernetes]]  
= VHM 및 Kubernetes

가상 호스트 관리자(VHM)을 사용하여 {kube} 클러스터를 관리할 수 있습니다.

VHM을 통해 {productname}는 클러스터에 대한 정보를 획득하고 보고할 수 있습니다. VHM에 대한 자세한 내용은 [xref:client-configuration:vhm.adoc\[ \]](#)을 참조하십시오 .

{productname}를 {kube}와 함께 사용하려면 컨테이너 관리를 위해 {productname}을 구성해야 합니다. 이와 함께 필요한 채널이 모두 있어야 하고 등록된 컨테이너 빌드 호스트를 사용할 수 있어야 합니다.

또한 다음 항목이 필요합니다 .

- \* 네트워크에서 사용할 수 있는 최소 한 개의 {kube} 클러스터
- \* {productname} 서버에 설치된 [systemitem]``virtual-host-gatherer-Kubernetes`` 패키지
- \* {kube} 버전 1.5.0 이상.
- \* 컨테이너 빌드 호스트의 Docker 버전 1.12 이상

**== {kube} VHM 생성**

{kube} 클러스터는 {productname}에서 VHM으로 등록됩니다.

{kube} 클러스터를 등록하고 인증하려면 ``kubeconfig`` 파일이 필요합니다. {kube} 명령줄 도구인 ``kubectl``을 사용해 ``kubeconfig`` 파일을 얻을 수 있습니다.

. 절차: {kube} VHM 생성

- . {productname} {webui}에서 menu:시스템[가상 호스트 관리자]로 이동합니다.
- . btn:[생성]을 클릭하고 [guimenu]``Kubernetes 클러스터``를 선택합니다.
- . [guimenu]``Kubernetes 가상 호스트 관리자 추가`` 섹션에서 다음과 같은 파라미터를 사용합니다.
  - \* [guimenu]``레이블`` 필드에 VHM의 사용자 정의 이름을 입력합니다.
  - \* {kube} 클러스터에 대한 필수 데이터를 포함하는 [path]``kubeconfig`` 파일을 선택합니다.
  - . [guimenu]``context`` 필드에서 클러스터에 적절한 컨텍스트를 선택합니다.
  - . [path]``kubeconfig`` 파일에 지정되어 있습니다.
- . btn:[생성]을 클릭합니다.

. 절차: 클러스터의 노드 보기

- . {productname} {webui}에서 menu:시스템[가상 호스트 관리자]로 이동합니다.
- . {kube} 클러스터를 선택합니다.
- . btn:[데이터 새로 고침 일정 잡기]를 클릭하여 노드 데이터를 새로 고침합니다.

노드 데이터가 업데이트되는 데 약간의 시간이 걸릴 수 있습니다. 업데이트된 정보를 보려면 브라우저 창을 새로 고침해야 할 수 있습니다.

모든 연결 또는 인증 문제는 [path]``gatherer.log``에 로깅됩니다.

#### [NOTE]

=====

등록 중에는 노드 데이터가 새로 고침되지 않습니다. 데이터는 수동으로 새로 고침해야 볼 수 있습니다.

=====

**== 이미지 런타임 데이터 검색**

{productname} {webui}에서 menu:이미지[이미지 목록]으로 이동하여 {kube} 이미지에 대한 런타임 데이터를 볼 수 있습니다.

이미지 목록 표에는 다음 세 가지 열이 포함되어 있습니다.

\* [guimenu] ``리비전`` :

+

{productname}이(가) 빌드한 이미지에 대한 모든 리빌드 또는 외부적으로 빌드되는 이미지에 대한 각 임포트에서 하나씩 증가하는 시퀀스 번호

\* [guimenu] ``런타임`` :

+

등록한 클러스터의 각 이미지에 대해 실행 중인 인스턴스의 전반적인 상태

\* [guimenu] ``인스턴스`` :

+

{productname}에 등록된 모든 클러스터에 걸쳐 이 이미지를 실행하는 인스턴스의 수 숫자 옆의 팝업 아이콘을 클릭하면 숫자를 상세히 확인할 수 있습니다.

[guimenu] ``런타임`` 열에는 다음 상태 메시지 중 하나가 표시됩니다.

\* ``모든 인스턴스가 SUSE Manager와 일치`` :

+

실행 중인 모든 인스턴스는 {productname}가 추적하는 이미지의 빌드와 동일한 빌드를 실행 중입니다.

\* ``오래된 인스턴스가 발견됨`` :

+

일부 인스턴스는 이미지의 이전 빌드를 실행 중입니다. 이미지를 다시 배포해야 할 수 있습니다.

\* ``정보 없음`` :

+

인스턴스 이미지의 체크섬은 {productname}에 포함된 이미지 데이터와 일치하지 않습니다.

이미지를 다시 배포해야 할 수 있습니다.

```
// This procedure needs help. LKB 2019-10-03
```

. 절차: 이미지 빌드

- . {productname} {webui}에서 menu:이미지[저장소]로 이동합니다.

- . btn:[생성]을 클릭하여 이미지 저장소를 생성합니다.

- . menu:이미지[프로파일]로 이동합니다.

- . btn:[생성]을 클릭하여 이미지 프로파일을 생성합니다.

  - {kube}에 배포하는 데 적합한 Dockerfile을 사용해야 합니다.

- . menu:이미지[빌드]로 이동하여 새 프로파일이 포함된 이미지를 빌드합니다.

- . 등록한 {kube} 클러스터 중 하나로 이미지를 배포합니다.

  - 이 작업은 [command] ``kubectl`` 도구로 할 수 있습니다.

업데이트된 데이터는 이제 menu:이미지[이미지 목록]의 이미지 목록에서 사용할 수 있어야 합니다.

```
// This procedure needs help. LKB 2019-10-03
```

- . 절차: 이전에 배포된 이미지 임포트
- . {productname} {webui}에서 menu:이미지[이미지 저장소]로 이동합니다.
- . 임포트하려는 이미지를 소유한 레지스트리를 추가합니다(아직 없는 경우).
- . menu:이미지[이미지 목록]으로 이동하여 btn:[임포트]를 클릭합니다.
- . 필드를 완성하고, 생성한 이미지 저장소를 선택하고, btn:[임포트]를 클릭합니다.

임포트한 이미지는 이제 menu:이미지[이미지 목록]의 이미지 목록에서 사용할 수 있어야 합니다.

- . 절차: 이전에 배포된 이미지 다시 빌드

- . {productname} {webui}에서 menu:이미지[이미지 목록]으로 이동하여 다시 빌드하려는 이미지가 포함된 행을 찾아 btn:[상세 정보]를 클릭합니다.
  - . [guimenu]``빌드 상태`` 섹션으로 이동하여 btn:[다시 빌드]를 클릭합니다.
- 다시 빌드하는 작업을 완료하는 데 시간이 다소 걸릴 수 있습니다.

다시 빌드하는 작업이 완료되면 menu:이미지[이미지 목록]의 이미지 목록에서 이미지의 런타임 상태가 업데이트됩니다. 이를 통해 인스턴스가 이미지의 이전 빌드를 실행 중임을 알 수 있습니다.

#### [NOTE]

=====

이미지가 원래 {productname}로 빌드된 경우에만 이미지를 다시 빌드할 수 있습니다. 임포트한 이미지는 다시 빌드할 수 없습니다.

=====

- . 절차: 추가 런타임 데이터 검색

- . {productname} {webui}에서 menu:이미지[이미지 목록]으로 이동하여 실행 중인 인스턴스가 포함된 행을 찾아 btn:[상세 정보]를 클릭합니다.
  - . [guimenu]``개요`` 탭으로 이동합니다.
- [guimenu]``이미지 정보`` 섹션의 [guimenu]``런타임`` 및 [guimenu]``인스턴스`` 필드에 데이터가 있습니다.
- . [guimenu]``런타임`` 탭으로 이동합니다.

이 섹션에는 등록한 모든 클러스터에서 이 이미지를 실행하는 {kube} Pod에 대한 정보가 포함되어 있습니다. 이 섹션의 정보는 다음 사항을 포함합니다.

+

- \* Pod 이름
- \* Pod가 상주하는 네임스페이스
- \* 특정 Pod에 있는 컨테이너의 런타임 상태

== 권한 및 인증서

**[IMPORTANT]**

=====

[path]``kubeconfig`` 파일은 모든 내장 인증서 데이터를 포함하는 경우에만 {productname}에서 사용할 수 있습니다.

=====

{productname}에서 실행하는 API 호출은 다음과 같습니다.

- \* ``GET /api/v1/pods``
- \* ``GET /api/v1/nodes``

{productname}에 대해 권장되는 최소 권한은 다음과 같습니다.

- \* 모든 노드를 나열할 수 있는 ClusterRole:
- +

resources: ["nodes"] verbs: ["list"]

\* 모든 네임스페이스에서 Pod를 나열할 수 있는 ClusterRole(역할 바인딩이 네임스페이스를 제한해서는 안 됨) :

+

리소스: [\pods\] verbs: [\list\]

``/pods``가 403 응답을 반환하면 {productname}가 전체 클러스터를 무시합니다.

RBAC 인증 작업에 대한 자세한 내용은

<https://kubernetes.io/docs/admin/authorization/rbac/> [ ]를 참조하십시오.

:leveloffset: 3  
 :leveloffset: +2

[[virt-nutanix]]  
 = Nutanix를 이용한 가상화

가상 호스트 관리자(VHM)를 설정하여 {productname}에서 Nutanix AHV 가상 머신을 사용할 수 있습니다. 먼저 {productname} 서버에서 VHM을 설정하고 사용 가능한 VM 호스트를 목록에 포함해야 합니다.

## == VHM 설정

가상 호스트 관리자(VHM)는 {productname} 서버에서 실행됩니다.

{productname} 서버에 [systemitem]``virtual-host-gatherer-Nutanix`` 패키지를 설치했는지 확인하십시오.

VHM을 실행하려면 {productname} 서버가 포트 9440을 열어 Nutanix Prism Element API에 액세스해야 합니다.

### . 절차: Nutanix VHM 생성

- . {productname} {webui}에서 menu:시스템[가상 호스트 관리자]로 이동합니다.
- . btn:[생성]을 클릭하고 [guimenu]``Nutanix AHV``를 선택합니다.
- . [guimenu]``Nutanix AHV 가상 호스트 관리자 추가`` 섹션에서 다음과 같은 파라미터를 사용합니다.
  - \* [guimenu]``레이블`` 필드에 VHM의 사용자 정의 이름을 입력합니다.
  - \* [guimenu]``호스트 이름`` 필드에 전체 도메인 이름(FQDN) 또는 호스트 IP 주소를 입력합니다.
  - \* [guimenu]``포트`` 필드에 사용할 Prism Element API 포트를 입력합니다(예: [parameter]``9440``).
  - \* [guimenu]``사용자 이름`` 필드에 VM 호스트와 연결된 사용자 이름을 입력합니다.
  - \* [guimenu]``사용자 이름`` 필드에 VM 호스트 사용자와 연결된 비밀번호를 입력합니다.
- . btn:[생성]을 클릭하여 변경 사항을 저장하고 VHM을 생성합니다.
- . [guimenu]``가상 호스트 관리자`` 페이지에서 새 VHM을 선택합니다.
- . [guimenu]``등록 정보`` 페이지에서 btn:[데이터 새로 고침]을 클릭하여 새 VHM을 목록에 포함합니다.

어떤 객체 및 리소스가 목록에 포함되었는지 확인하려면 menu:시스템[시스템 목록 > 가상 시스템]으로 이동하십시오.

### [NOTE]

=====

HTTPS를 사용해 브라우저에서 Nutanix Prism API 서버에 연결하면 때로 ``잘못된 인증서`` 오류가 로깅될 수 있습니다. 이런 오류가 발생하는 경우 가상 호스트 관리자에서 데이터를 새로 고침하는 작업이 실패합니다. 유효한 SSL 인증서(자체 서명되지 않음)가 Nutanix API 서버에 필요합니다. Nutanix SSL 인증서에 대해 사용자 정의 CA(인증 주체)를 사용 중인 경우 사용자 정의 CA 인증서를 {productname} 서버의 [path]``/etc/pki/trust/anchors``로 복사하십시오. 명령줄에서 [command]``update-ca-certificates`` 명령을 실행하여 인증서를 다시 신뢰하고 spacewalk 서비스를 다시 시작합니다.

=====

taskomatic은 VHM을 생성하여 구성한 후 데이터 수집을 자동으로 실행합니다. 데이터 수집을

수동으로 수행하려면 menu:시스템[가상 호스트 관리자]로 이동하여 적절한 VHM을 선택한 후 btn:[데이터 새로 고침]을 클릭합니다.

{productname}는 API를 사용해 VHM에 연결하고 가상 호스트에 대한 정보를 요청할 수 있는 [command]``virtual-host-gatherer``라고 하는 도구와 함께 제공됩니다. [command]``virtual-host-gatherer``는 각 모듈이 특정 VHM을 활성화하는 옵션 모듈의 개념을 유지합니다. 이 도구는 Taskomatic이 야간에 자동으로 호출합니다. [command]``virtual-host-gatherer`` 도구의 로그 파일은 [path]``/var/log/rhn/gatherer.log``에 있습니다.

```
:leveloffset: 3
:leveloffset: +2
```

```
[[virt-vmware]]
= VMWare를 이용한 가상화
```

가상 호스트 관리자(VHM)를 설정하여 {productname}에서 ESXi 및 vCenter를 포함한 VMWare vSphere 가상 머신을 사용할 수 있습니다.

먼저 {productname} 서버에서 VHM을 설정하고 사용 가능한 VM 호스트를 목록에 포함해야 합니다. 이렇게 하면 Taskomatic이 VM API를 사용해 데이터 수집을 시작할 수 있습니다.

== VHM 설정

가상 호스트 관리자(VHM)는 {productname} 서버에서 실행됩니다.

VHM을 실행하려면 {productname} 서버가 포트 443을 열어 VMWare API에 액세스해야 합니다.

VMWare 호스트는 액세스 역할 및 권한을 사용해 호스트 및 게스트에 대한 액세스를 제어합니다. VHM에 의해 목록에 포함되었으면 하는 모든 VMWare 객체 또는 리소스에 최소한 [parameter]``읽기 전용`` 권한이 있는지 확인합니다. 객체 또는 리소스를 제외하려면 해당 객체 또는 리소스를 [parameter]``no-access``로 표시하십시오.

새 호스트를 {productname}에 추가하려면 사용자 및 객체에 할당된 역할 및 권한을 {productname}가 목록에 포함해야 할지 여부를 고려해야 합니다.

사용자, 역할 및 권한에 대한 자세한 내용은 VMWare vSphere 문서([https://docs.vmware.com/en/VMware-vSphere/index.html\[\]](https://docs.vmware.com/en/VMware-vSphere/index.html[]))에서 참조하십시오.

. 절차: VMWare VHM 생성

- . {productname} {webui}에서 menu:시스템[가상 호스트 관리자]로 이동합니다.
- . btn:[생성]을 클릭하고 [guimenu]``VMWare 기반``을 선택합니다.

- . [guimenu] ``VMWare 기반 가상 호스트 관리자 추가`` 섹션에서 다음과 같은 파라미터를 사용합니다.
  - \* [guimenu] ``레이블`` 필드에 VHM의 사용자 정의 이름을 입력합니다.
  - \* [guimenu] ``호스트 이름`` 필드에 전체 도메인 이름(FQDN) 또는 호스트 IP 주소를 입력합니다.
  - \* [guimenu] ``포트`` 필드에 사용할 ESXi API 포트를 입력합니다(예: [parameter] ``443``).
  - \* [guimenu] ``사용자 이름`` 필드에 VM 호스트와 연결된 사용자 이름을 입력합니다.
  - \* [guimenu] ``사용자 이름`` 필드에 VM 호스트 사용자와 연결된 비밀번호를 입력합니다.
- . btn:[생성]을 클릭하여 변경 사항을 저장하고 VHM을 생성합니다.
- . [guimenu] ``가상 호스트 관리자`` 페이지에서 새 VHM을 선택합니다.
- . [guimenu] ``등록 정보`` 페이지에서 btn:[데이터 새로 고침]을 클릭하여 새 VHM을 목록에 포함합니다.

어떤 객체 및 리소스가 목록에 포함되었는지 확인하려면 menu:시스템[시스템 목록 > 가상 시스템]으로 이동하십시오.

#### [NOTE]

=====

HTTPS를 사용해 브라우저에서 ESXi 서버에 연결하면 때로 ``잘못된 인증서`` 오류가 로깅될 수 있습니다. 이런 오류가 발생하는 경우 가상 호스트 서버에서 데이터를 새로 고침하는 작업이 실패합니다. 이 문제를 해결하려면 ESXi 서버에서 인증서를 추출하여 [path] ``/etc/pki/trust/anchors``로 복사하십시오. 명령줄에서 [command] ``update-ca-certificates`` 명령을 실행하여 인증서를 다시 신뢰하고 spacewalk 서비스를 다시 시작합니다.

=====

taskomatic은 VHM을 생성하여 구성한 후 데이터 수집을 자동으로 실행합니다. 데이터 수집을 수동으로 수행하려면 menu:시스템[가상 호스트 관리자]로 이동하여 적절한 VHM을 선택한 후 btn:[데이터 새로 고침]을 클릭합니다.

{productname}는 API를 사용해 VHM에 연결하고 가상 호스트에 대한 정보를 요청할 수 있는 [command] ``virtual-host-gatherer``라고 하는 도구와 함께 제공됩니다.

[command] ``virtual-host-gatherer``는 각 모듈이 특정 VHM을 활성화하는 옵션 모듈의 개념을 유지합니다. 이 도구는 Taskomatic 야간에 자동으로 호출합니다.

[command] ``virtual-host-gatherer`` 도구의 로그 파일은

[path] ``/var/log/rhn/gatherer.log``에 있습니다.

== VMWare에서 SSL 오류 문제 해결

VMWare 설치를 구성하는 중에 SSL 오류가 발생한 경우 VMWare에서 CA 인증서 파일을 다운로드하여 {productname}에서 신뢰해야 합니다.

- . 절차: VMWare CA 인증서 신뢰
- . VMWare 설치에서 CA 인증서를 다운로드합니다.
  - vCenter {webui}에 로그인해 btn:[신뢰할 수 있는 루트 CA 인증서 다운로드]를 클릭하여 다운로드할 수 있습니다.
  - . 다운로드한 CA 인증서 파일이 ``.zip`` 형식인 경우 압축을 풉니다.  
인증서 파일의 확장자는 숫자입니다. 예를 들면 ``certificate.0``과 같습니다.
  - . 인증서 파일을 {productname} 서버로 복사하고 [path]``/etc/pki/trust/anchors/`` 디렉토리에 저장합니다.
  - . 복사한 인증서의 파일 이름 접미사를 ``.crt`` 또는 ``.pem``으로 변경합니다.
  - . {productname} 서버의 명령 프롬프트에서 다음과 같이 CA 인증서 레코드를 업데이트합니다.
- +

#### update-ca-certificates

```
:leveloffset: 3
[leveloffset: +2
```

#### [[virt-file]]

= 다른 타사 공급자를 통한 가상화

Xen, KVM 또는 VMware가 아닌 타사 가상화 공급자를 사용하려면 JSON 구성 파일을 {productname}로 임포트하면 됩니다.

이와 마찬가지로 API에 대한 직접 액세스를 제공하지 않는 VMWare가 설치된 경우 파일 기반 VHM은 몇 가지 기본 관리 기능을 제공합니다.

#### [NOTE]

====

이 옵션은 [command]``virtual-host-gatherer`` 도구로 생성한 파일을 임포트하기 위한 것입니다. 이 옵션은 수동으로 생성한 파일을 위해 설계된 것은 아닙니다.

=====

```
// Add instructions for custom JSON file, including example if possible.
LKB 2019-10-23
// https://github.com/uyuni-project/uyuni-rfc/blob/master/accepted/00056-
subscription-matching-in-public-clouds.md#the-output-from-a-virtual-host-
gatherer-plugin
```

#### . 절차: JSON 파일 엑스포트 및 임포트

- . VM 네트워크에서 [command]``virtual-host-gatherer``를 실행하여 JSON 구성 파일을 엑스포트합니다.
- . 생성한 파일을 {productname} 서버가 액세스할 수 있는 위치에 저장합니다.
- . {productname} {webui}에서 menu:시스템[가상 호스트 관리자]로 이동합니다.

- . btn:[생성]을 클릭하고 [guimenu]``파일 기반``을 선택합니다.
- . [guimenu]``파일 기반 가상 호스트 관리자 추가`` 섹션에서 다음과 같은 파라미터를 사용합니다.
  - \* [guimenu]``레이블`` 필드에 VHM의 사용자 정의 이름을 입력합니다.
  - \* [guimenu]``Url`` 필드에 엑스포트한 JSON 구성 파일의 경로를 입력합니다.
- . btn:[생성]을 클릭하여 변경 사항을 저장하고 VHM을 생성합니다.
- . [guimenu]``가상 호스트 관리자`` 페이지에서 새 VHM을 선택합니다.
- . [guimenu]``등록 정보`` 페이지에서 btn:[데이터 새로 고침]을 클릭하여 새 VHM을 목록에 포함합니다.

. 예: 엑스포트한 JSON 구성 파일:

```
{
 "examplevhost": {
 "10.11.12.13": {
 "cpuArch": "x86_64",
 "cpuDescription": "AMD Opteron(tm) Processor 4386",
 "cpuMhz": 3092.212727,
 "cpuVendor": "amd",
 "hostIdentifier": "vim.HostSystem:host-182",
 "name": "11.11.12.13",
 "os": "VMware ESXi",
 "osVersion": "5.5.0",
 "ramMb": 65512,
 "totalCpuCores": 16,
 "totalCpuSockets": 2,
 "totalCpuThreads": 16,
 "type": "vmware",
 "vms": {
 "vCenter": "564d6d90-459c-2256-8f39-3cb2bd24b7b0"
 }
 },
 "10.11.12.14": {
 "cpuArch": "x86_64",
 "cpuDescription": "AMD Opteron(tm) Processor 4386",
 "cpuMhz": 3092.212639,
 "cpuVendor": "amd",
 "hostIdentifier": "vim.HostSystem:host-183",
 "name": "10.11.12.14",
 "os": "VMware ESXi",
 "osVersion": "5.5.0",
 "ramMb": 65512,
 "totalCpuCores": 16,
 "totalCpuSockets": 2,
 "totalCpuThreads": 16,
 "type": "vmware",
 "vms": {
 "49737e0a-c9e6-4ceb-aef8-6a9452f67cb5": "4230c60f-3f98-2a65-f7c3-600b26b79c22",
 "5a2e4e63-a957-426b-bfa8-4169302e4fdb": "42307b15-1618-0595-01f2-427ffcd88e",
 "NSX-gateway": "4230d43e-aafe-38ba-5a9e-3cb67c03a16a",
 "NSX-l3gateway": "4230b00f-0b21-0e9d-dfde-6c7b06909d5f",
 "NSX-service": "4230e924-b714-198b-348b-25de01482fd9"
 }
 }
 }
}
```

자세한 내용은 {productname} 서버의 사용자 지정 페이지에서 [command]``virtual-host-gatherer``를 검색하십시오.

man virtual-host-gatherer

이 패키지의 `README` 파일은 하이퍼바이너의 ‘유형’ 등에 대한 배경 정보를 제공합니다.

/usr/share/doc/packages/virtual-host-gatherer/README.md

사용자 지정 페이지 및 ‘README’ 파일에는 예시 구성 파일도 포함되어 있습니다.

```
:leveloffset: 3
:leveloffset: +1
= GNU Free Documentation License
Copyright (C) 2000, 2001, 2002 Free Software Foundation, Inc.
```

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

```
[float]
== 0. PREAMBLE
```

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially.

Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense.

It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does.

But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book.

We recommend this License principally for works whose purpose is instruction or reference.

```
[float]
== 1. APPLICABILITY AND DEFINITIONS
```

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License.

Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein.

The "Document", below, refers to any such manual or work.

Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with

modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject.

(Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant.

The Document may contain zero Invariant Sections.

If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters.

A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent.

An image format is not Transparent if used for any substantial amount of text.

A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification.

Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page.

For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language.

(Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document.

These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

```
[float]
== 2. VERBATIM COPYING
```

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License.

You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute.

However, you may accept compensation in exchange for copies.

If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

```
[float]
== 3. COPYING IN QUANTITY
```

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover.

Both covers must also clearly and legibly identify you as the publisher of these copies.

The front cover must present the full title with all words of the title equally prominent and visible.

You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material.

If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

```
[float]
== 4. MODIFICATIONS
```

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the

Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

[upperalpha]

- . Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- . List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- . State on the Title page the name of the publisher of the Modified Version, as the publisher.
- . Preserve all the copyright notices of the Document.
- . Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- . Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- . Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- . Include an unaltered copy of this License.
- . Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- . Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- . For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

- . Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- . Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- . Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- . Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant.

To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice.

These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version.

Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity.

If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

```
[float]
== 5. COMBINING DOCUMENTS
```

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list

them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number.

Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

[float]  
== 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

[float]  
== 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit.

When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies

of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

[float]  
== 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections.

You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers.

In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

[float]  
== 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License.

Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

[float]  
== 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time.

Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation.

If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

[float]

-- ADDENDUM: How to use this License for your documents

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the {ldquo} with...Texts.{rdquo} line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

:leveloffset: 3