# Red Hat Enterprise Linux 9

## systemd 장치 파일을 사용하여 시스템 사용자 지정 및 최적화

시스템 성능 최적화 및 systemd를 사용하여 구성 확장

Last Updated: 2024-06-26

# Red Hat Enterprise Linux 9 systemd 장치 파일을 사용하여 시스템 사용자 지정 및 최적화

시스템 성능 최적화 및 systemd를 사용하여 구성 확장

## 법적 공지



## 초록

systemd 장치 파일을 수정하고 기본 구성을 확장하고 시스템 부팅 성능을 검사하고 systemd를 최적화하여 부팅 시간을 단축합니다.

# 차례

# RED HAT 문서에 관한 피드백 제공

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

**Jira를 통해 피드백 제출 (등록 필요)**

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

# 1장. SYSTEMD 장치 파일 작업

**systemd** 장치 파일은 시스템 리소스를 나타냅니다. 시스템 관리자는 다음과 같은 고급 작업을 수행할 수 있습니다.

- 사용자 지정 단위 파일 생성
- 기존 장치 파일 수정
- 인스턴스화된 단위 작업

## 1.1. 단위 파일 소개

단위 파일에는 장치를 설명하고 해당 동작을 정의하는 구성 지시문이 포함되어 있습니다. 여러 **systemctl** 명령은 백그라운드에서 유닛 파일에서 작동합니다. 보다 세밀하게 조정하려면 수동으로 단위 파일을 편집하거나 생성할 수 있습니다. 유닛 파일이 시스템에 저장되는 세 가지 기본 디렉터리를 찾을 수 있습니다. **/etc/systemd/system/** 디렉터리는 시스템 관리자가 만들거나 사용자 지정하는 단위 파일용으로 예약되어 있습니다.

단위 파일 이름은 다음과 같은 형태를 취합니다.

> *<unit_name>.<type_extension>*

여기서 *unit_name* 은 단위 이름을 나타내며 *type_extension* 는 단위 유형을 식별합니다.

예를 들어 시스템에 있는 **sshd.socket** 유닛과 **sshd.service** 장치를 찾을 수 있습니다.

추가 구성 파일을 위한 디렉터리로 단위 파일을 추가할 수 있습니다. 예를 들어 **sshd.service** 에 사용자 지정 구성 옵션을 추가하려면 **sshd.service.d/custom.conf** 파일을 생성하고 여기에 추가 지시문을 삽입합니다. 구성 디렉터리에 대한 자세한 내용은 기존 장치 파일 수정을 참조하십시오.

**systemd** 시스템 및 서비스 관리자는 **sshd.service.wants/** 및 **sshd.service.requires/** 디렉터리를 생성할 수도 있습니다. 이러한 디렉터리에는 **sshd** 서비스의 종속 항목인 단위 파일에 대한 심볼릭 링크가 포함되어 있습니다. **systemd** 는 [Install] 장치 파일 옵션에 따라 설치 중 또는 [Unit] 옵션을 기반으로 런타임 시 심볼릭 링크를 자동으로 생성합니다. 이러한 디렉터리 및 심볼릭 링크를 수동으로 만들 수도 있습니다.

또한 **sshd.service.wants/** 및 **sshd.service.requires/** 디렉터리를 생성할 수 있습니다. 이러한 디렉터리에는 **sshd** 서비스의 종속 항목인 단위 파일에 대한 심볼릭 링크가 포함되어 있습니다. 심볼릭 링크는 설치 중에 [Install] 단위 파일 옵션 또는 런타임에 [Unit] 옵션을 기반으로 자동으로 생성됩니다. 이러한 디렉터리와 심볼릭 링크를 수동으로 만들 수도 있습니다. [Install] 및 [Unit] 옵션에 대한 자세한 내용은 아래 표를 참조하십시오.

많은 단위 파일 옵션은 단위 파일이 로드될 때 **장치** 매개 변수로 동적으로 대체되는 와일드카드 문자열을 사용하여 설정할 수 있습니다. 이를 통해 인스턴스화된 단위를 생성하기 위한 템플릿 역할을 하는 일반 단위 파일을 생성할 수 있습니다. 인스턴스화된 유닛 작업을 참조하십시오.

## 1.2. SYSTEMD 장치 파일 위치

다음 디렉토리 중 하나에서 단위 구성 파일을 찾을 수 있습니다.

**표 1.1. systemd 장치 파일 위치**

| 디렉터리 | 설명 |
| --- | --- |
| **/usr/lib/systemd/system/** | 설치된 RPM 패키지와 함께 배포된 **systemd** 장치 파일. |
| **/run/systemd/system/** | 런타임에 생성된 **systemd** 장치 파일입니다. 이 디렉터리는 설치된 서비스 단위 파일이 있는 디렉토리보다 우선합니다. |
| **/etc/systemd/system/** | **systemctl enable** 명령과 서비스 확장을 위해 추가된 유닛 파일을 사용하여 생성된 **systemd** 장치 파일입니다. 이 디렉터리는 런타임 단위 파일이 있는 디렉터리보다 우선합니다. |

**systemd** 의 기본 구성은 컴파일 중에 정의되며 **/etc/systemd/system.conf** 파일에서 구성을 찾을 수 있습니다. 이 파일을 편집하여 전역적으로 **systemd** 단위의 값을 재정의하여 기본 구성을 수정할 수 있습니다.

예를 들어 90초로 설정된 시간 제한의 기본값을 재정의하려면 **DefaultTimeoutStartSec** 매개변수를 사용하여 필요한 값을 초 단위로 입력합니다.

```
DefaultTimeoutStartSec=required value
```

## 1.3. 단위 파일 구조

단위 파일은 일반적으로 다음 세 섹션으로 구성됩니다.

**[Unit] 섹션**

단위 유형에 의존하지 않는 일반 옵션을 포함합니다. 이러한 옵션은 단위 설명을 제공하고, 유닛의 동작을 지정하고, 종속성을 다른 단위로 설정합니다. 가장 자주 사용되는 [Unit] 옵션 목록은 중요한 [Unit] 섹션 옵션을 참조하십시오.

**[Unit type] 섹션**

유형별 지시문을 포함합니다. 단위 유형 뒤에 이름이 지정된 섹션에서 그룹화됩니다. 예를 들어 서비스 단위 파일에는 **[Service]** 섹션이 포함되어 있습니다.

**[Install] 섹션**

**systemctl enable** 및 **disable** 명령에서 사용하는 장치 설치에 대한 정보를 포함합니다. **[Install]** 섹션의 옵션 목록은 중요한 [설치] 섹션을 참조하십시오.

**추가 리소스**

- 중요한 [Unit] 섹션 옵션
- 중요한 [서비스] 섹션 옵션
- 중요 [설치] 섹션 옵션

## 1.4. 중요한 [UNIT] 섹션 옵션

다음 표에는 [Unit] 섹션의 중요한 옵션이 나열되어 있습니다.

**표 1.2. 중요한 [Unit] 섹션 옵션**

| 옵션 [a] | 설명 |
|---|---|
| **설명** | 단위에 대한 의미 있는 설명입니다. 이 텍스트는 예를 들어 **systemctl status** 명령의 출력에 표시됩니다. |
| **문서** | 단위에 대한 문서를 참조하는 URI 목록을 제공합니다. |
| **이후**[b] | 단위가 시작되는 순서를 정의합니다. 단위는 에 지정된 단위 후에만 시작됩니다. **Requires 와 달리after 은 지정된 장치를 명시적으로 활성화하지 않습니다. 이전 옵션에 는 뒤에 대한 반대 기능이 있습니다.** |
| **필요** | 다른 유닛에 대한 종속성을 구성합니다. **Requires** 에 나열된 단위는 유닛과 함께 활성화됩니다. 필요한 유닛을 시작하지 못하면 장치가 활성화되지 않습니다. |
| **want** | **Requires** 보다 약한 종속성을 구성합니다. 나열된 장치가 성공적으로 시작되지 않으면 장치 활성화에 영향을 미치지 않습니다. 이는 사용자 지정 단위 종속 항목을 설정하는 데 권장되는 방법입니다. |
| **충돌** | **Requires** 와 반대로 음수 종속성을 구성합니다. |

[a] [Unit] 섹션에서 구성 가능한 전체 옵션 목록은 **systemd.unit(5)** 매뉴얼 페이지를 참조하십시오.

[b] 대부분의 경우 **After** 및 **Before** 단위 파일 옵션을 사용하여 순서 종속 항목만 설정하는 것으로 충분합니다. 또한 **Wants** (권장) 또는 **Requires** 를 사용하여 요구 사항 종속성을 설정하는 경우 순서 종속성을 계속 지정해야 합니다. 이는 순서 지정 및 요구 사항 종속성이 서로 독립적으로 작동하기 때문입니다.

## 1.5. 중요한 [서비스] 섹션 옵션

다음 표에는 [Service] 섹션의 중요한 옵션이 나열되어 있습니다.

**표 1.3. 중요한 [서비스] 섹션 옵션**

| 옵션 [a] | 설명 |
|---|---|

| 옵션 [a] | 설명 |
| --- | --- |
| **유형** | **ExecStart** 및 관련 옵션 기능에 영향을 주는 단위 프로세스 시작 유형을 구성합니다. 그 중 하나:<br><br>* **simple** - 기본값입니다. **ExecStart** 로 시작된 프로세스는 서비스의 주요 프로세스입니다.<br><br>시작 **- ExecStart** 로 시작된 프로세스는 서비스의 주요 프로세스가 되는 하위 프로세스를 생성합니다. 상위 프로세스는 시작이 완료되면 종료됩니다.<br><br>* **shot** - 이 유형은 **단순** 과 유사하지만 연속 단위를 시작하기 전에 프로세스가 종료됩니다.<br><br>* **dbus** - 이 유형은 **단순** 과 유사하지만 기본 프로세스가 D-Bus 이름을 얻은 후에만 연속 단위가 시작됩니다.<br><br>* **통지** - 이 유형은 **단순** 과 유사하지만 sd_notify() 함수를 통해 알림 메시지를 보낸 후에만 연속 단위가 시작됩니다.<br><br>**유휴** - **단순** 과 유사하게 모든 작업이 완료될 때까지 서비스 바이너리의 실제 실행이 지연되므로 상태 출력을 서비스의 쉘 출력과 혼합하지 않습니다. |
| **ExecStart** | 단위가 시작될 때 실행할 명령 또는 스크립트를 지정합니다. **ExecStartPre** 및 **ExecStartPost** 는 **ExecStart** 전후에 실행할 사용자 지정 명령을 지정합니다. **type=oneshot** 을 사용하면 순차적으로 실행되는 여러 사용자 지정 명령을 지정할 수 있습니다. |
| **ExecStop** | 단위가 중지될 때 실행할 명령 또는 스크립트를 지정합니다. |
| **ExecReload** | 장치를 다시 로드할 때 실행할 명령 또는 스크립트를 지정합니다. |
| **재시작** | 이 옵션을 활성화하면 **systemctl** 명령에서 clean stop 을 제외하고 프로세스가 종료된 후 서비스가 다시 시작됩니다. |
| **RemainAfterExit** | True로 설정하면 모든 프로세스가 종료된 경우에도 서비스가 활성 상태로 간주됩니다. 기본값은 False입니다. 이 옵션은 **Type=oneshot** 이 구성된 경우 특히 유용합니다. |
| [a] [Service] 섹션에서 구성 가능한 전체 옵션 목록은 **systemd.service(5)** 매뉴얼 페이지를 참조하십시오. | |

## 1.6. 중요 [설치] 섹션 옵션

다음 표에는 [Install] 섹션의 중요한 옵션이 나열되어 있습니다.

표 1.4. 중요 [설치] 섹션 옵션

| 옵션 [a] | 설명 |
|---|---|
| **별칭** | 단위에 대한 공백으로 구분된 추가 이름 목록을 제공합니다. **systemctl enable** 를 제외한 대부분의 systemctl 명령은 실제 단위 이름 대신 별칭을 사용할 수 있습니다. |
| **RequiredBy** | 단위에 종속된 단위 목록입니다. 이 장치를 활성화하면 **RequiredBy** 에 나열된 단위는 유닛에 **대한** 요구 사항을 얻습니다. |
| **WantedBy** | 유닛에 약한 단위 목록입니다. 이 장치를 활성화하면 **WantedBy** 에 나열된 단위는 단위에 **대한** 종속성을 얻습니다. |
| **또한** | 단위와 함께 설치 또는 제거할 단위 목록을 지정합니다. |
| **DefaultInstance** | 인스턴스화된 단위로 제한되는 이 옵션은 단위가 활성화된 기본 인스턴스를 지정합니다. 인스턴스화된 유닛 작업을 참조하십시오. |
| [a] [Install] 섹션에서 구성 가능한 전체 옵션 목록은 **systemd.unit(5)** 매뉴얼 페이지를 참조하십시오. | |

## 1.7. 사용자 지정 단위 파일 생성

처음부터 단위 파일을 만드는 몇 가지 사용 사례가 있습니다. 사용자 지정 데몬을 실행하고 sshd 서비스의 두 번째 인스턴스를 사용하여 사용자 지정 단위 파일 생성에서 로 일부 기존 서비스의 두 번째 인스턴스를만들 수 있습니다.

반면 기존 유닛의 동작을 수정하거나 확장하려는 경우 기존 장치 파일 수정 의 지침을 사용하십시오.

**프로세스**

1. 사용자 지정 서비스를 생성하려면 서비스로 실행 파일을 준비합니다. 파일에는 사용자 정의 생성된 스크립트 또는 소프트웨어 공급자가 제공하는 실행 파일이 포함될 수 있습니다. 필요한 경우 사용자 지정 서비스의 기본 프로세스에 대해 일정한 PID를 보유하도록 PID 파일을 준비합니다. 환경 파일을 포함하여 서비스의 쉘 변수를 저장할 수도 있습니다. **chmod a+x**를 실행하여 소스 스크립트가 실행 가능하고 대화형이 아닌지 확인합니다.

2. **/etc/systemd/system/** 디렉터리에 유닛 파일을 만들고 올바른 파일 권한이 있는지 확인합니다. **root** 로 실행 :

```
# touch /etc/systemd/system/<name>.service

# chmod 664 /etc/systemd/system/<name>.service
```

& *lt;name* >을 생성할 서비스 이름으로 교체합니다. 파일을 실행할 필요가 없습니다.

3. 생성된 &lt ***;name>*** **.service** 파일을 열고 서비스 구성 옵션을 추가합니다. 생성하려는 서비스 유형에 따라 다양한 옵션을 사용할 수 있습니다. 단위 파일 구조를 참조하십시오.
다음은 네트워크 관련 서비스에 대한 단위 구성의 예입니다.

```
[Unit]
Description=<service_description>
After=network.target

[Service]
ExecStart=<path_to_executable>
Type=forking
PIDFile=<path_to_pidfile>

[Install]
WantedBy=default.target
```

- *<service_description* >은 저널 로그 파일과 **systemctl status** 명령의 출력에 표시되는 정보를 제공합니다.
- **After** 설정을 사용하면 네트워크가 실행된 후에만 서비스가 시작됩니다. 공백으로 구분된 기타 관련 서비스 또는 대상 목록을 추가합니다.
- *path_to_executable* 은 실제 서비스 실행 파일의 경로를 나타냅니다.
- **type=forking** 은 포크 시스템 호출을 수행하는 데몬에 사용됩니다. 서비스의 기본 프로세스는 *path_to_pidfile* 에 지정된 PID를 사용하여 생성됩니다. 중요한 [Service] 섹션 옵션에서 다른 시작 유형을 찾습니다.
- **WantedBy** 는 서비스를 시작해야 하는 대상 또는 대상을 지정합니다. 이러한 목표를 이전 수준의 개념을 대체하도록 고려하십시오.

4. 새 < ***name>*** **.service** 파일이 있음을 **systemd** 에 알립니다.

```
# systemctl daemon-reload

# systemctl start <name>.service
```

주의

새 장치 파일을 생성하거나 기존 장치 파일을 수정한 후 항상 **systemctl daemon-reload** 명령을 실행합니다. 그렇지 않으면 **systemd** 상태와 실제 서비스 단위 파일의 상태가 일치하지 않아 **systemctl start** 또는 **systemctl enable** 명령이 실패할 수 있었습니다. 단위가 많은 시스템에서 각 장치의 상태를 직렬화하고 나중에 다시 로드하는 동안 역직렬화해야 하므로 시간이 오래 걸릴 수 있습니다.

## 1.8. SSHD 서비스의 두 번째 인스턴스를 사용하여 사용자 지정 단위 파일 생성

서비스 인스턴스를 여러 개 구성하고 실행해야 하는 경우 원래 서비스 구성 파일의 복사본을 생성하고 특정 매개변수를 수정하여 서비스의 기본 인스턴스와의 충돌을 방지할 수 있습니다.

**프로세스**

**sshd** 서비스의 두 번째 인스턴스를 생성하려면 다음을 수행합니다.

1. 두 번째 데몬이 사용할 **sshd_config** 파일의 사본을 생성합니다.

```
# cp /etc/ssh/sshd{,-second}_config
```

2. 이전 단계에서 만든 **sshd-second_config** 파일을 편집하여 다른 포트 번호와 PID 파일을 두 번째 데몬에 할당합니다.

```
Port 22220
PidFile /var/run/sshd-second.pid
```

**Port** 및 **PidFile** 옵션에 대한 자세한 내용은 **sshd_config**(5) 매뉴얼 페이지를 참조하십시오. 선택한 포트가 다른 서비스에서 사용하지 않는지 확인합니다. PID 파일은 서비스를 실행하기 전에 존재할 필요가 없으며 서비스 시작 시 자동으로 생성됩니다.

3. **sshd** 서비스에 대한 **systemd** 장치 파일의 사본을 생성합니다.

```
# cp /usr/lib/systemd/system/sshd.service /etc/systemd/system/sshd-second.service
```

4. 생성된 **sshd-second.service** 를 변경합니다.

   a. **설명** 옵션을 수정합니다.

```
Description=OpenSSH server second instance daemon
```

   b. 첫 번째 인스턴스가 이미 시작된 후에만 두 번째 인스턴스가 시작되도록 **After** 옵션에 지정된 서비스에 **sshd.service** 를 추가합니다.

```
After=syslog.target network.target auditd.service sshd.service
```

   c. sshd의 첫 번째 인스턴스에는 키 생성이 포함된 **ExecStartPre=/usr/sbin/ sshd -keygen** 행을 제거합니다.

   d. 대체 구성 파일이 사용되도록 **-f /etc/ssh/ sshd -second_config** 매개변수를 sshd 명령에 추가합니다.

```
ExecStart=/usr/sbin/sshd -D -f /etc/ssh/sshd-second_config $OPTIONS
```

   e. 수정 후 **sshd-second.service** 장치 파일에는 다음 설정이 포함됩니다.

```
[Unit]
Description=OpenSSH server second instance daemon
After=syslog.target network.target auditd.service sshd.service

[Service]
EnvironmentFile=/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D -f /etc/ssh/sshd-second_config $OPTIONS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s
```

```
[Install]
WantedBy=multi-user.target
```

5. SELinux를 사용하는 경우 **sshd** 의 두 번째 인스턴스의 포트를 SSH 포트에 추가합니다. 그렇지 않으면 **sshd** 의 두 번째 인스턴스가 포트에 바인딩됩니다.

```
# semanage port -a -t ssh_port_t -p tcp 22220
```

6. 부팅 시 **sshd-second.service** 를 자동으로 시작합니다.

```
# systemctl enable sshd-second.service
```

7. **systemctl status** 명령을 사용하여 **sshd-second.service** 가 실행 중인지 확인합니다.

8. 서비스에 연결하여 포트가 올바르게 활성화되어 있는지 확인합니다.

```
$ ssh -p 22220 user@server
```

**sshd** 의 두 번째 인스턴스에 대한 연결을 허용하도록 방화벽을 구성해야 합니다.

## 1.9. SYSTEMD 서비스 설명 찾기

**#description** 로 시작하는 줄에서 스크립트에 대한 설명 정보를 찾을 수 있습니다. 단위 파일의 [Unit] 섹션에 있는 **Description** (설명) 옵션에서 이 설명을 서비스 이름과 함께 사용합니다. 헤더에는 **#Short-Description** 및 **#Description** 행에 대한 유사한 데이터가 포함될 수 있습니다.

## 1.10. SYSTEMD 서비스 종속 항목 찾기

Linux 표준 기본(LSB) 헤더에는 서비스 간의 종속성을 형성하는 여러 지시문이 포함될 수 있습니다. 대부분은 systemd 장치 옵션으로 변환이 가능합니다. 다음 표를 참조하십시오.

**표 1.5. LSB 헤더의 종속성 옵션**

| LSB 옵션 | 설명 | 단위 파일 동등성 |
|---|---|---|
| **제공** | 다른 init 스크립트에서 "$" 접두사를 사용하여 참조할 수 있는 서비스의 부팅 기능 이름을 지정합니다. 단위 파일이 파일 이름으로 다른 단위를 참조하므로 더 이상 필요하지 않습니다. | – |
| **required-Start** | 필수 서비스의 부팅 기능 이름을 포함합니다. 이는 순서 지정 종속성으로 변환되며 부팅 기능 이름이 속한 해당 서비스 또는 대상의 단위 파일 이름으로 교체됩니다. 예를 들어 **postfix** 의 경우 $network에 대한 Required-Start 종속성이 network.target에 대한 After 종속성으로 변환되었습니다. | **후 후,전에** |

| LSB 옵션 | 설명 | 단위 파일 동등성 |
|---|---|---|
| **should-Start** | Required-Start보다 약한 종속 항목을 구성합니다. failed should-Start 종속 항목은 서비스 시작에 영향을 미치지 않습니다. | **후 후,전에** |
| **required-Stop,should-Stop** | 부정적인 종속 항목을 구성합니다. | **충돌** |

## 1.11. 서비스의 기본 대상 찾기

**#chkconfig** 로 시작하는 행에는 세 개의 숫자 값이 포함되어 있습니다. 가장 중요한 것은 서비스가 시작되는 기본 실행 수준을 나타내는 첫 번째 번호입니다. 이러한 level을 해당하는 systemd 대상에 매핑합니다. 그런 다음 단위 파일의 [Install] 섹션에 있는 **WantedBy** 옵션에 이러한 대상을 나열합니다. 예를 들어, **postfix** 는 이전에 multi-user.target 및 graphical.target으로 변환되는 잠정식 2, 3, 4, 5에서 시작되었습니다. graphical.target은 multiuser.target에 따라 달라지므로 둘 다 지정할 필요가 없습니다. LSB 헤더의 **#Default-Start** 및 **#Default-Stop** 줄에서도 기본 및 금지된 level에 대한 정보를 찾을 수 있습니다.

**#chkconfig** 행에 지정된 나머지 두 값은 init 스크립트의 시작 및 종료 우선순위를 나타냅니다. 이러한 값은 init 스크립트를 로드하는 경우 **systemd** 에서 해석되지만 이와 동등한 단위 파일이 없습니다.

## 1.12. 서비스에서 사용하는 파일 검색

init 스크립트를 사용하려면 전용 디렉터리에서 함수 라이브러리를 로드하고 구성, 환경 및 PID 파일을 가져올 수 있습니다. 환경 변수는 **EnvironmentFile** 단위 파일 옵션으로 변환되는 init 스크립트 헤더에서 **#config** 로 시작하는 줄에 지정됩니다. **#pidfile** init 스크립트 행에 지정된 PID 파일은 **PIDFile** 옵션을 사용하여 단위 파일로 가져옵니다.

init 스크립트 헤더에 포함되지 않은 주요 정보는 서비스 실행 파일의 경로이며 서비스에 필요한 기타 파일일 수 있습니다. 이전 버전의 Red Hat Enterprise Linux에서 init 스크립트는 Bash case 문을 사용하여 **시작,중지** 또는 **재시작** 과 같은 기본 작업에 대한 서비스 동작을 정의했습니다. **postfix** init 스크립트의 다음 발췌 내용은 서비스 시작 시 실행할 코드 블록을 보여줍니다.

```
conf_check() {
    [ -x /usr/sbin/postfix ] || exit 5
    [ -d /etc/postfix ] || exit 6
    [ -d /var/spool/postfix ] || exit 5
}

make_aliasesdb() {
 if [ "$(/usr/sbin/postconf -h alias_database)" == "hash:/etc/aliases" ]
 then
  # /etc/aliases.db might be used by other MTA, make sure nothing
  # has touched it since our last newaliases call
  [ /etc/aliases -nt /etc/aliases.db ] ||
   [ "$ALIASESDB_STAMP" -nt /etc/aliases.db ] ||
   [ "$ALIASESDB_STAMP" -ot /etc/aliases.db ] || return
  /usr/bin/newaliases
  touch -r /etc/aliases.db "$ALIASESDB_STAMP"
 else
  /usr/bin/newaliases
 fi
```

```
}

start() {
 [ "$EUID" != "0" ] && exit 4
 # Check that networking is up.
 [ ${NETWORKING} = "no" ] && exit 1
 conf_check
 # Start daemons.
 echo -n $"Starting postfix: "
 make_aliasesdb >/dev/null 2>&1
 [ -x $CHROOT_UPDATE ] && $CHROOT_UPDATE
 /usr/sbin/postfix start 2>/dev/null 1>&2 && success || failure $"$prog start"
 RETVAL=$?
 [ $RETVAL -eq 0 ] && touch $lockfile
        echo
 return $RETVAL
}
```

init 스크립트의 확장성을 통해 **start()** 함수 블록에서 호출되는 두 가지 사용자 지정 함수 **conf_check()** 및 **make_aliasesdb()** 를 지정할 수 있었습니다. 자세히 살펴보면 위의 코드에서 몇 개의 외부 파일과 디렉터리가 언급됩니다. 기본 서비스 실행 파일 **/usr/sbin/** Cryostat , **/etc/ Cryostat/** 및 **/var/spool/ Cryostat/** 구성 디렉토리와 **/usr/sbin/postconf/** 디렉터리입니다.

**systemd** 는 사전 정의된 작업만 지원하지만 ExecStart , **ExecStartPre**, **ExecStart Post**,**ExecStop**, **ExecReload** 옵션을 사용하여 사용자 지정 실행 파일을 실행할 수 있습니다. 지원 스크립트와 함께 **/usr/sbin/** Cryostat가 서비스 시작 시 실행됩니다. 복잡한 init 스크립트를 변환하려면 스크립트의 모든 문의 목적을 이해해야 합니다. 일부 설명은 운영 체제 버전에 고유하므로 번역할 필요가 없습니다. 반면 단위 파일뿐만 아니라 서비스 실행 파일 및 지원 파일에서 새 환경에 약간의 조정이 필요할 수 있습니다.

## 1.13. 기존 장치 파일 수정

기존 장치 파일을 수정하려면 **/etc/systemd/system/** 디렉터리로 이동합니다. 시스템이 **/usr/lib/systemd/system/** 디렉터리에 저장하는 기본 단위 파일을 수정하지 않아야 합니다.

**프로세스**

1. 필요한 변경의 범위에 따라 다음 방법 중 하나를 선택합니다.

   - **/etc/systemd/system/ <*unit*> .d/** 에 보조 구성 파일의 디렉터리를 만듭니다. 이 방법은 대부분의 사용 사례에 권장됩니다. 원래 장치 파일을 계속 참조하면서 추가 기능을 사용하여 기본 구성을 확장할 수 있습니다. 따라서 패키지 업그레이드와 함께 도입된 기본 단위 변경 사항이 자동으로 적용됩니다. 자세한 내용은 기본 단위 구성 확장을 참조하십시오.

   - **'/ etc/systemd/system/ 디렉터리에 있는 /usr/lib/systemd/system/'directory에서** 원본 유닛 파일의 사본을 만들고 변경합니다. 복사본은 원본 파일을 재정의하므로 패키지 업데이트에 도입된 변경 사항은 적용되지 않습니다. 이 방법은 패키지 업데이트에 관계없이 유지되어야 하는 중요한 단위 변경을 수행하는 데 유용합니다. 자세한 내용은 기본 단위 구성 덮어쓰기를 참조하십시오.

2. 단위의 기본 구성으로 돌아가려면 **/etc/systemd/system/** 디렉터리에 있는 사용자 지정 생성 구성 파일을 삭제합니다.

3. 시스템을 재부팅하지 않고 단위 파일에 변경 사항을 적용합니다.

   ```
   # systemctl daemon-reload
   ```

**daemon-reload** 옵션은 모든 장치 파일을 다시 로드하고 전체 종속성 트리를 다시 생성합니다. 이 트리는 모든 변경 사항을 단위 파일에 즉시 적용하는 데 필요합니다. 또는 다음 명령을 사용하여 동일한 결과를 얻을 수 있습니다.

```
# init q
```

4. 수정된 장치 파일이 실행 중인 서비스에 속하는 경우 서비스를 다시 시작하십시오.

```
# systemctl restart <name>.service
```

중요

SysV initscript에서 처리하는 서비스의 종속성 또는 타임아웃과 같은 속성을 수정하려면 initscript 자체를 수정하지 마십시오. 대신 에 설명된 대로 서비스에 대한 **systemd** 드롭인 구성 파일을 생성합니다. 기본 장치 구성 확장 및 기본 장치 구성 덮어쓰기 .

그런 다음 일반 **systemd** 서비스와 동일한 방식으로 이 서비스를 관리합니다.

예를 들어 **네트워크** 서비스의 구성을 확장하려면 **/etc/rc.d/init.d/network** initscript 파일을 수정하지 마십시오. 대신 새 디렉토리 **/etc/systemd/system/network.service.d/** 및 **systemd** 드롭인 파일 **/etc/systemd/system/network.service.d/*my_config*.conf** 를 만듭니다. 그런 다음 수정된 값을 드롭인 파일에 넣습니다. 참고: **systemd** 는 **네트워크** 서비스를 **network.service** 로 알고 있으므로 생성된 디렉터리를 **network.service.d**라고 해야 합니다.

## 1.14. 기본 단위 구성 확장

추가 systemd 구성 옵션을 사용하여 기본 장치 파일을 확장할 수 있습니다.

**프로세스**

1. **/etc/systemd/system/**:에 구성 디렉토리를 만듭니다.

```
# mkdir /etc/systemd/system/<name>.service.d/
```

& *lt;name* >을 확장하려는 서비스 이름으로 바꿉니다. 구문은 모든 단위 유형에 적용됩니다.

2. **.conf** 접미사를 사용하여 구성 파일을 생성합니다.

```
# touch /etc/systemd/system/name.service.d/<config_name>.conf
```

& *lt;config_name*& gt;을 구성 파일의 이름으로 바꿉니다. 이 파일은 일반 장치 파일 구조를 준수하고 해당 섹션의 모든 지시문을 지정해야 합니다. 단위 파일 구조를 참조하십시오.

예를 들어 사용자 정의 종속성을 추가하려면 다음 콘텐츠를 사용하여 구성 파일을 생성합니다.

```
[Unit]
Requires=<new_dependency>
After=<new_dependency>
```

& *lt;new_dependency* >는 장치가 종속성으로 표시되도록 나타냅니다. 또 다른 예로는 기본 프로세스가 종료된 후 30초가 지연된 후 서비스를 다시 시작하는 구성 파일이 있습니다.

```
[Service]
Restart=always
RestartSec=30
```

하나의 작업에만 중점을 둔 작은 구성 파일을 생성합니다. 이러한 파일은 쉽게 이동하거나 다른 서비스의 구성 디렉터리에 연결할 수 있습니다.

3. 단위에 변경 사항을 적용합니다.

```
# systemctl daemon-reload
# systemctl restart <name>.service
```

**예 1.1. httpd.service 구성 확장**

Apache 서비스를 시작할 때 사용자 지정 쉘 스크립트가 자동으로 실행되도록 **httpd.service** 장치를 수정하려면 다음 단계를 수행합니다.

1. 디렉터리 및 사용자 지정 구성 파일을 생성합니다.

```
# mkdir /etc/systemd/system/httpd.service.d/
```

```
# touch /etc/systemd/system/httpd.service.d/custom_script.conf
```

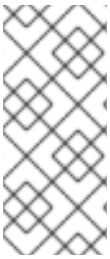
2. **custom_script.conf** 파일에 다음 텍스트를 삽입하여 기본 서비스 프로세스 후에 실행할 스크립트를 지정합니다.

```
[Service]
ExecStartPost=/usr/local/bin/custom.sh
```

3. 단위 변경 사항을 적용합니다.

```
# systemctl daemon-reload
```

```
# systemctl restart httpd.service
```



**참고**

**/etc/systemd/system/** 구성 디렉터리의 구성 파일이 **/usr/lib/systemd/system/** 의 단위 파일보다 우선합니다. 따라서 구성 파일에 **Description** 또는 **ExecStart** 와 같이 한 번만 지정할 수 있는 옵션이 포함된 경우 이 옵션의 기본값이 재정의됩니다. Monitoring (모니터링)에 설명된 **systemd-delta** 명령의 출력에서 이러한 단위는 합계에 있어도 항상 [EXTENDED]로 표시됩니다.

## 1.15. 기본 단위 구성 덮어쓰기

장치 파일을 제공하는 패키지를 업데이트한 후 유지할 단위 파일 구성을 변경할 수 있습니다.

**프로세스**

1. **root** 로 다음 명령을 입력하여 장치 파일을 **/etc/systemd/system/** 디렉터리에 복사합니다.

```
# cp /usr/lib/systemd/system/<name>.service /etc/systemd/system/<name>.service
```

2. 텍스트 편집기로 복사된 파일을 열고 변경합니다.

3. 단위 변경 적용:

```
# systemctl daemon-reload
# systemctl restart <name>.service
```

## 1.16. 시간 제한 변경

서비스당 시간 초과 값을 지정하여 오작동 서비스가 시스템 정지되지 않도록 할 수 있습니다. 그렇지 않으면 일반 서비스의 경우 시간 제한 기본값은 90초, SysV 호환 서비스의 경우 300초입니다.

**프로세스**

**httpd** 서비스의 제한 시간을 확장하려면 다음을 수행합니다.

1. **httpd** 유닛 파일을 **/etc/systemd/system/** 디렉터리에 복사합니다.

```
# cp /usr/lib/systemd/system/httpd.service /etc/systemd/system/httpd.service
```

2. **/etc/systemd/system/httpd.service** 파일을 열고 **[Service]** 섹션에 **TimeoutStartUSec** 값을 지정합니다.

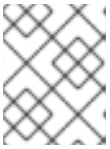
```
...
[Service]
...
PrivateTmp=true
TimeoutStartSec=10

[Install]
WantedBy=multi-user.target
...
```

3. **systemd** 데몬을 다시 로드합니다.

```
# systemctl daemon-reload
```

4. **선택 사항:** 새 시간 초과 값을 확인합니다.

```
# systemctl show httpd -p TimeoutStartUSec
```



**참고**

제한 시간 제한을 전역적으로 변경하려면 **/etc/systemd/system.conf** 파일에 **DefaultTimeoutStartSec** 을 입력합니다.

## 1.17. 재정의된 단위 모니터링

**systemd-delta** 명령을 사용하여 재정의되거나 수정된 유닛 파일의 개요를 표시할 수 있습니다.

**프로세스**

- 재정의되거나 수정된 단위 파일의 개요를 표시합니다.

```
# systemd-delta
```

예를 들어 명령의 출력은 다음과 같습니다.

```
[EQUIVALENT] /etc/systemd/system/default.target → /usr/lib/systemd/system/default.target
[OVERRIDDEN] /etc/systemd/system/autofs.service →
/usr/lib/systemd/system/autofs.service

--- /usr/lib/systemd/system/autofs.service      2014-10-16 21:30:39.000000000 -0400
+++ /etc/systemd/system/autofs.service  2014-11-21 10:00:58.513568275 -0500
@@ -8,7 +8,8 @@
 EnvironmentFile=-/etc/sysconfig/autofs
 ExecStart=/usr/sbin/automount $OPTIONS --pid-file /run/autofs.pid
 ExecReload=/usr/bin/kill -HUP $MAINPID
-TimeoutSec=180
+TimeoutSec=240
+Restart=Always

 [Install]
 WantedBy=multi-user.target

[MASKED]    /etc/systemd/system/cups.service → /usr/lib/systemd/system/cups.service
[EXTENDED]   /usr/lib/systemd/system/sssd.service →
/etc/systemd/system/sssd.service.d/journal.conf

4 overridden configuration files found.
```

## 1.18. 인스턴스화된 단위 작업

단일 템플릿 구성을 사용하여 서비스 인스턴스를 여러 개 관리할 수 있습니다. 단위에 대한 일반 템플릿을 정의하고 런타임 시 특정 매개 변수를 사용하여 해당 단위의 여러 인스턴스를 생성할 수 있습니다. 템플릿은 at 기호(@)로 표시됩니다. 인스턴스화된 단위는 다른 단위 파일( **필수** 또는 **Wants** 옵션 사용) 또는 **systemctl start** 명령을 사용하여 시작할 수 있습니다. 인스턴스화된 서비스 단위의 이름은 다음과 같습니다.

```
<template_name>@<instance_name>.service
```

& *lt;template_name* >은 템플릿 구성 파일의 이름을 나타냅니다. & *lt;instance_name*& gt;을 단위 인스턴스의 이름으로 바꿉니다. 유닛의 모든 인스턴스에 공통 구성 옵션을 사용하여 동일한 템플릿 파일을 가리킬 수 있습니다. 템플릿 단위 이름의 형식은 다음과 같습니다.

```
<unit_name>@.service
```

예를 들어 다음 **Wants** 설정은 단위 파일에서 설정됩니다.

```
Wants=getty@ttyA.service getty@ttyB.service
```

먼저 systemd가 지정된 서비스 유닛을 검색합니다. 이러한 단위를 찾을 수 없는 경우 "@"과 type 접미사 사이의 부분이 무시되고 **systemd** 는 **getty@.service** 파일을 검색하고 해당 장치에서 구성을 읽고 서비스를 시작합니다.

예를 들어 **getty@.service** 템플릿에는 다음 지시문이 포함되어 있습니다.

```
[Unit]
Description=Getty on %I
...
[Service]
ExecStart=-/sbin/agetty --noclear %I $TERM
...
```

위의 템플릿에서 **getty@ttyA.service** 및 **getty@ttyB.service** 가 인스턴스화되면 description =이 **ttyA에서 Getty로 확인되고 tty B의 Getty** 로 확인됩니다.

## 1.19. 중요한 단위 지정자

**단위 구성 파일에서 단위 지정자 라고 하는 와일드카드 문자를 사용할 수 있습니다. 단위 지정자는 특정 단위 매개변수를 대체하며 런타임 시 해석됩니다.**

**표 1.6. 중요한 단위 지정자**

| 단위 지정자 | 의미 | 설명 |
|---|---|---|
| **%n** | 전체 단위 이름 | type 접미사를 포함한 전체 단위 이름을 나타냅니다. **%** n은 의미가 동일하지만 금지된 문자를 ASCII 코드로 대체합니다. |
| **%p** | 접두사 이름 | 접미사가 제거된 단위 이름을 나타냅니다. 인스턴스화된 단위의 경우 %p는 "@" 문자 앞에 있는 단위 이름의 부분을 나타냅니다. |
| **%i** | 인스턴스 이름 | "@" 문자와 유형 접미사 사이의 인스턴스화된 단위 이름의 일부입니다. **%I** 는 동일한 의미를 가지지만 ASCII 코드에 금지된 문자도 대체합니다. |
| **%H** | 호스트 이름 | 단위 구성이 로드되는 시점에 실행 중인 시스템의 호스트 이름을 나타냅니다. |
| **%t** | 런타임 디렉터리 | **root** 사용자에 대한 **/run** 또는 권한이 없는 사용자의 XDG_RUNTIME_DIR 변수 값인 런타임 디렉터리를 나타냅니다. |

**단위 지정자의 전체 목록은 systemd.unit(5) 매뉴얼 페이지를 참조하십시오.**

### 1.20. 추가 리소스

- RHEL 및 systemd에서 서비스 제한 설정 방법
- 특정 서비스를 시작해야 하는 서비스 단위 파일을 작성하는 방법
- systemd 서비스 장치 정의가 있어야 하는 종속성을 결정하는 방법

# 2장. SYSTEMD를 최적화하여 부팅 시간 단축

시스템 관리자는 시스템의 성능을 최적화하고 부팅 시간을 단축할 수 있습니다. 부팅 중에 **systemd** 가 시작되는 서비스를 검토하고 필요한 사항을 평가할 수 있습니다. 특정 서비스가 부팅 시 시작되도록 비활성화하면 시스템의 부팅 시간이 개선될 수 있습니다.

## 2.1. 시스템 부팅 성능 검사

시스템 부팅 성능을 검사하려면 **systemd-analyze** 명령을 사용할 수 있습니다. 특정 옵션을 사용하면 **systemd**를 조정하여 부팅 시간을 단축할 수 있습니다.

사전 요구 사항

- 선택 사항: **systemd** 를 검사하여 부팅 시간을 조정하기 전에 활성화된 모든 서비스를 나열합니다.

```
$ systemctl list-unit-files --state=enabled
```

프로세스

분석할 정보를 선택합니다.

- 마지막으로 성공적으로 부팅한 시간에 대한 정보를 분석합니다.

```
$ systemd-analyze
```

- 각 **systemd** 장치의 단위 초기화 시간을 분석합니다.

```
$ systemd-analyze blame
```

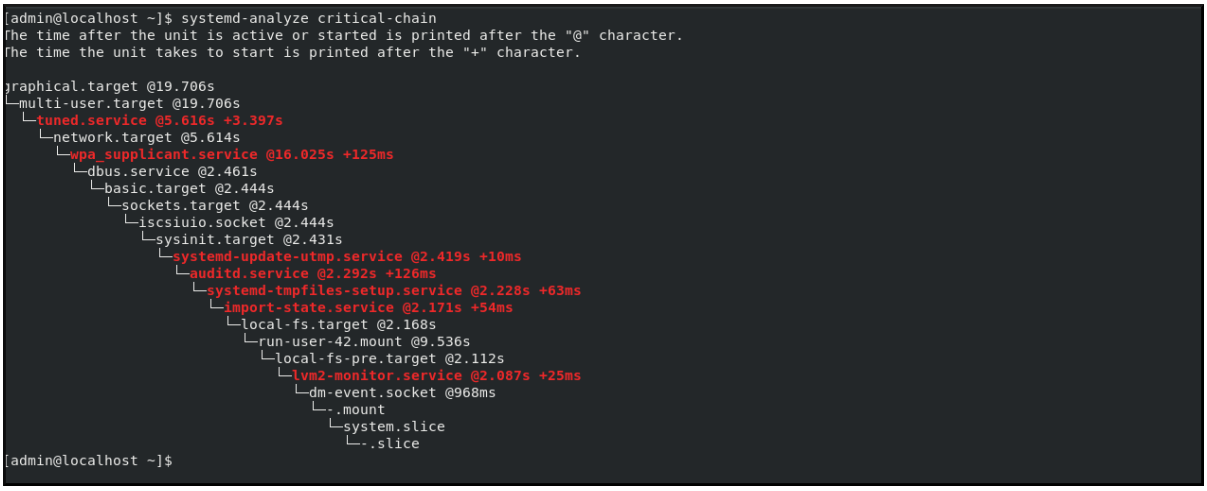
마지막 성공적인 부팅 중에 초기화하는 데 걸린 시간에 따라 단위가 내림차순으로 나열됩니다.

- 마지막 성공적인 부팅 시 초기화하는 데 시간이 오래 걸리는 중요한 단위를 확인합니다.

```
$ systemd-analyze critical-chain
```

출력에는 빨간색 색상으로 부팅 속도가 크게 느려지는 단위가 강조 표시됩니다.

그림 2.1. systemd-analyze critical-chain 명령 출력




추가 리소스

- systemd-analyze(1) 도움말 페이지

## 2.2. 안전하게 비활성화할 수 있는 서비스 선택 가이드

기본적으로 부팅 시 활성화된 특정 서비스를 비활성화하여 시스템의 부팅 시간을 단축할 수 있습니다.

- 활성화된 서비스를 나열합니다.

```
$ systemctl list-unit-files --state=enabled
```

- 서비스를 비활성화합니다.

```
# systemctl disable <service_name>
```

특정 서비스는 운영 체제가 안전하고 필요한 방식으로 작동하도록 활성화되어야 합니다.

안전하게 비활성화할 수 있는 서비스를 선택하는 가이드로 다음 표를 참조하십시오. 표에는 Red Hat Enterprise Linux의 최소 설치 시 기본적으로 활성화된 모든 서비스가 나열되어 있습니다.

표 2.1. RHEL의 최소 설치에서 기본적으로 활성화된 서비스

| 서비스 이름 | 비활성화될 수 있습니까? | 더 많은 정보 |
|---|---|---|
| auditd.service | 제공됨 | 커널의 감사 메시지가 필요하지 않은 경우에만 **auditd.service** 를 비활성화합니다. **auditd.service** 를 비활성화하는 경우 **/var/log/audit/audit.log** 파일이 생성되지 않습니다. 결과적으로 사용자 로그인, 서비스 시작 또는 암호 변경과 같은 일반적으로 검토된 작업 또는 이벤트를 소급할 수 없습니다. 또한 auditd에는 커널 부분과 서비스 자체의 두 부분이 있습니다. **systemctl disable auditd** 명령을 사용하면 커널 부분이 아닌 서비스만 비활성화합니다. 전체 시스템 감사를 비활성화하려면 커널 명령줄에서 **audit=0** 을 설정합니다. |
| autovt@.service | 제공되지 않음 | 이 서비스는 실제로 필요한 경우에만 실행되므로 비활성화할 필요가 없습니다. |
| crond.service | 제공됨 | crond.service를 비활성화하면 crontab의 항목이 실행되지 않습니다. |
| dbus-org.fedoraproject.FirewallD1.service | 제공됨 | **firewalld.service**에 대한 심볼릭 링크 |
| dbus-org.freedesktop.NetworkManager.service | 제공됨 | **NetworkManager.service**에 대한 심볼릭 링크 |
| dbus-org.freedesktop.nm-dispatcher.service | 제공됨 | **NetworkManager-dispatcher.service**의 심볼릭 링크 |
| firewalld.service | 제공됨 | 방화벽이 필요하지 않은 경우에만 **firewalld.service** 를 비활성화합니다. |
| getty@.service | 제공되지 않음 | 이 서비스는 실제로 필요한 경우에만 실행되므로 비활성화할 필요가 없습니다. |
| import-state.service | 제공됨 | 네트워크 스토리지에서 부팅할 필요가 없는 경우에만 **import-state.service** 를 비활성화합니다. |
| irqbalance.service | 제공됨 | CPU가 하나만 있는 경우에만 **irqbalance.service** 를 비활성화합니다. 여러 CPU가 있는 시스템에서 **irqbalance.service** 를 비활성화하지 마십시오. |
| kdump.service | 제공됨 | 커널 충돌에서 보고서가 필요하지 않은 경우에만 **kdump.service** 를 비활성화합니다. |
| loadmodules.service | 제공됨 | **/etc/rc.modules** 또는 **/etc/sysconfig/modules** 디렉터리가 존재하지 않는 한 이 서비스는 시작되지 않습니다. 즉, 최소 RHEL 설치에서 시작되지 않습니다. |

| 서비스 이름 | 비활성화될 수 있습니까? | 더 많은 정보 |
|---|---|---|
| lvm2-monitor.service | 제공됨 | LVM(Logical Volume Manager)을 사용하지 않는 경우에만 **lvm2-monitor.service** 를 비활성화합니다. |
| microcode.service | 제공되지 않음 | CPU에서 마이크로 코드 소프트웨어의 업데이트를 제공하므로 서비스를 비활성화하지 마십시오. |
| NetworkManager-dispatcher.service | 제공됨 | 네트워크 구성 변경에 대한 알림이 필요하지 않은 경우에만 **NetworkManager-dispatcher.service** 를 비활성화합니다(예: 정적 네트워크). |
| NetworkManager-wait-online.service | 제공됨 | 부팅 직후 사용 가능한 작업 네트워크 연결이 필요하지 않은 경우에만 **NetworkManager-wait-online.service** 를 비활성화합니다. 서비스가 활성화되면 네트워크 연결이 작동하기 전에 시스템이 부팅을 완료하지 않습니다. 이로 인해 부팅 시간이 크게 증가할 수 있습니다. |
| NetworkManager.service | 제공됨 | 네트워크에 연결할 필요가 없는 경우에만 **NetworkManager.service** 를 비활성화합니다. |
| nis-domainname.service | 제공됨 | NIS(Network Information Service)를 사용하지 않는 경우에만 **nis-domainname.service** 를 비활성화합니다. |
| rhsmcertd.service | 제공되지 않음 | |
| rngd.service | 제공됨 | 시스템에 엔트로피가 필요하지 않거나 일종의 하드웨어 생성기가 없는 경우에만 **rngd.service** 를 비활성화합니다. 서비스는 X.509 인증서 생성에 사용되는 시스템과 같이 좋은 엔트로피가 많은 환경에서 필요합니다(예: FreeIPA 서버). |
| rsyslog.service | 제공됨 | 영구 로그가 필요하지 않은 경우에만 **rsyslog.service** 를 비활성화하거나 **systemd-journald** 를 영구 모드로 설정합니다. |
| selinux-autorelabel-mark.service | 제공됨 | SELinux를 사용하지 않는 경우에만 **selinux-autorelabel-mark.service** 를 비활성화합니다. |
| sshd.service | 제공됨 | OpenSSH 서버에서 원격 로그인이 필요하지 않은 경우에만 **sshd.service** 를 비활성화합니다. |
| sssd.service | 제공됨 | 네트워크를 통해 시스템에 로그인하는 사용자가 없는 경우에만 **sssd.service** 를 비활성화합니다(예: LDAP 또는 Kerberos 사용). **sssd.service** 를 비활성화하는 경우 모든 **sssd-*** 장치를 비활성화하는 것이 좋습니다. |
| syslog.service | 제공됨 | **rsyslog.service** 의 별칭 |

| 서비스 이름 | 비활성화될 수 있습니까? | 더 많은 정보 |
|---|---|---|
| tuned.service | 제공됨 | 성능 튜닝을 사용해야 하는 경우에만 **tuned.service** 를 비활성화합니다. |
| lvm2-lvmpolld.socket | 제공됨 | LVM(Logical Volume Manager)을 사용하지 않는 경우에만 **lvm2-lvmpolld.socket** 을 비활성화합니다. |
| dnf-makecache.timer | 제공됨 | 패키지 메타데이터를 자동으로 업데이트할 필요가 없는 경우에만 **dnf-makecache.timer** 를 비활성화합니다. |
| unbound-anchor.timer | 제공됨 | DNSSEC(DNS Security Extensions)에 대한 루트 신뢰 앵커를 매일 업데이트할 필요가 없는 경우에만 **unbound-anchor.timer** 를 비활성화합니다. 이 루트 신뢰 앵커는 DNSSEC 검증을 위해 Unbound resolver 및 resolver 라이브러리에서 사용됩니다. |

**서비스에 대한 자세한 정보를 찾으려면 다음 명령 중 하나를 사용합니다.**

```
$ systemctl cat <service_name>
```

```
$ systemctl help <service_name>
```

**systemctl cat** 명령은 각 **/usr/lib/systemd/system/ <service >** 서비스 파일과 적용되는 모든 덮어쓰기의 콘텐츠를 제공합니다. 적용 가능한 덮어쓰기에는 **/etc/systemd/system/ <service >** 파일의 장치 파일 덮어쓰기 또는 해당 **device.type.d** 디렉터리의 드롭인 파일이 포함됩니다.

추가 리소스

- **systemd.unit(5)** 도움말 페이지
- 특정 서비스의 도움말 페이지를 표시하는 **systemd help** 명령

## 2.3. 추가 리소스

- **systemctl(1)** 도움말 페이지
- **systemd(1)** 도움말 페이지

- **systemd-delta(1)** 도움말 페이지
- **systemd.directives(7)** 도움말 페이지
- **systemd.unit(5)** 도움말 페이지
- **systemd.service(5) man page**
- **systemd.target(5) man page**
- **systemd.kill(5)** 도움말 페이지
- **systemd** 홈 페이지