



Red Hat Enterprise Linux 9

Web サーバーとリバースプロキシのデプロイ

Red Hat Enterprise Linux 9 に Web サーバーとリバースプロキシをデプロイするためのガイド

Red Hat Enterprise Linux 9 Web サーバーとリバースプロキシのデプロイ

Red Hat Enterprise Linux 9 に Web サーバーとリバースプロキシをデプロイするためのガイド

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Deploying_web_servers_and_reverse_proxies.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

このドキュメントでは、Red Hat Enterprise Linux 9 で Web サーバーとプロキシサーバー (Apache HTTP サーバー、NGINX、および Squid) を設定して実行する方法について説明します。

目次

多様性を受け入れるオープンソースの強化	3
RED HAT ドキュメントへのフィードバック (英語のみ)	4
第1章 APACHE HTTP WEB サーバーの設定	5
1.1. APACHE HTTP WEB サーバーの概要	5
1.2. APACHE HTTP SERVER への主な変更点	5
1.3. APACHE 設定ファイル	6
1.4. HTTPD サービスの管理	6
1.5. シングルインスタンスの APACHE HTTP サーバーの設定	7
1.6. APACHE 名ベースの仮想ホストの設定	8
1.7. APACHE HTTP WEB サーバーの KERBEROS 認証の設定	10
1.7.1. IdM 環境で GSS-Proxy の設定	10
1.7.2. Apache HTTP Web サーバーが共有するディレクトリーに Kerberos 認証の設定	11
1.8. APACHE HTTP サーバーで TLS 暗号化の設定	12
1.8.1. Apache HTTP Server への TLS 暗号化の追加	12
1.8.2. Apache HTTP サーバーでサポートされる TLS プロトコルバージョンの設定	14
1.8.3. Apache HTTP サーバーで対応している暗号の設定	15
1.9. TLS クライアント証明書認証の設定	16
1.10. APACHE HTTP SERVER のマニュアルのインストール	17
1.11. モジュールの使用	18
1.11.1. モジュールの読み込み	18
1.11.2. モジュールの作成	18
1.12. APACHE WEB サーバー設定で秘密鍵と証明書を使用できるように NSS データベースからの証明書のエクスポート	19
1.13. 関連情報	19
第2章 NGINX の設定および構成	20
2.1. NGINX のインストールおよび準備	20
2.2. 異なるドメインに異なるコンテンツを提供する WEB サーバーとしての NGINX の設定	21
2.3. NGINX WEB サーバーへの TLS 暗号化の追加	23
2.4. HTTP トラフィックのリバースプロキシとしての NGINX の設定	25
2.5. NGINX を HTTP ロードバランサーとして設定	25
2.6. 関連情報	26
第3章 SQUID キャッシングプロキシサーバーの設定	27
3.1. 認証なしで SQUID をキャッシングプロキシとして設定	27
3.2. LDAP 認証を使用したキャッシングプロキシとしての SQUID の設定	29
3.3. KERBEROS 認証を使用したキャッシングプロキシとしての SQUID の設定	32
3.4. SQUID でのドメイン拒否リストの設定	36
3.5. SQUID サービスが特定のポートまたは IP アドレスをリッスンするように設定	36
3.6. 関連情報	37

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#)をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

ご意見ご要望をお聞かせください。ドキュメントの改善点はございませんか。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
 1. ドキュメントの表示が **Multi-page HTML** 形式になっていて、ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
 2. マウスカーソルで、コメントを追加する部分を強調表示します。
 3. そのテキストの下に表示される **Add Feedback** ポップアップをクリックします。
 4. 表示される手順に従ってください。
- Bugzilla を介してフィードバックを送信するには、新しいチケットを作成します。
 1. [Bugzilla](#) の Web サイトに移動します。
 2. Component で **Documentation** を選択します。
 3. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも記入してください。
 4. **Submit Bug** をクリックします。

第1章 APACHE HTTP WEB サーバーの設定

1.1. APACHE HTTP WEB サーバーの概要

Web サーバーは、Web 経由でクライアントにコンテンツを提供するネットワークサービスです。これは通常 Web ページを指しますが、他のドキュメントも当てはまります。Web サーバーは、ハイパーテキスト転送プロトコル (HTTP) を使用するため、HTTP サーバーとも呼ばれます。

Apache HTTP Server (**httpd**) は、[Apache Software Foundation](#) が開発したオープンソースの Web サーバーです。

Red Hat Enterprise Linux の以前のリリースからアップグレードする場合は、適切に **httpd** サービス設定を更新する必要があります。本セクションでは、新たに追加された機能の一部と、以前の設定ファイルの更新を説明します。

1.2. APACHE HTTP SERVER への主な変更点

RHEL 9 は、Apache HTTP Server のバージョン 2.4.48 を提供します。RHEL 8 に同梱されるバージョン 2.4.37 からの変更には、以下が含まれます。

- Apache HTTP Server Control Interface (**apachectl**):
 - **apachectl status** 出力では、**systemctl** ページャーが無効になりました。
 - 追加の引数を渡すと警告が表示される代わりに、**apachectl** コマンドが失敗するようになりました。
 - **apachectl graceful-stop** がすぐに戻るようになりました。
 - **apachectl configtest** コマンドが、SELinux コンテキストを変更せずに、**httpd -t** コマンドを実行するようになりました。
 - RHEL の **apachectl(8)** man ページで、アップストリームの **apachectl** との相違点が完全に説明されるようになりました。
- Apache eXtenSion ツール (**apxs**):
 - **/usr/bin/apxs** コマンドは、**httpd** パッケージのビルド時に適用されたコンパイラーの最適化フラグを使用または公開しなくなりました。**/usr/lib64/httpd/build/vendor-apxs** コマンドを使用して、**httpd** のビルドに使用されるのと同じコンパイラーフラグを適用できるようになりました。**vendor-apxs** コマンドを使用するには、最初に **redhat-rpm-config** パッケージをインストールする必要があります。
- Apache モジュール:
 - **mod_lua** モジュールが、別のパッケージで提供されるようになりました。
- 設定構文の変更
 - **mod_access_compat** が提供する非推奨の **Allow** ディレクティブでは、コメント (**#** 文字) が暗黙的に無視される代わりにシンタックスエラーを発生するようになりました。
- その他の変更:
 - カーネルスレッド ID は、エラーログメッセージで直接使用されるようになり、精度と簡潔性が向上しました。

- 多くのマイナーな機能強化とバグ修正
- モジュールの作成者には、利用可能な新しいインターフェースが多数用意されています。

RHEL 8 以降、**httpd** モジュールAPI に後方互換性のない変更はありません。

Apache HTTP Server 2.4 は、この Apache HTTP Server 2.4 の初期バージョンです。これは、RPM パッケージとして簡単にインストールできます。

1.3. APACHE 設定ファイル

httpd サービスが起動すると、デフォルトでは、[表1.1 「httpd サービスの設定ファイル」](#) に記載されている場所から設定が読み込まれます。

表1.1 httpd サービスの設定ファイル

パス	詳細
<code>/etc/httpd/conf/httpd.conf</code>	主要設定ファイル。
<code>/etc/httpd/conf.d/</code>	主要設定ファイル内に含まれている設定ファイル用の補助ディレクトリー。
<code>/etc/httpd/conf.modules.d/</code>	Red Hat Enterprise Linux にパッケージ化されたインストール済みの動的モジュールを読み込む設定ファイルの補助ディレクトリー。デフォルト設定では、この設定ファイルが最初に処理されます。

デフォルト設定はほとんどの状況に適していますが、その他の設定オプションを使用することもできます。設定の変更を有効にするには、Web サーバーを再起動します。**httpd** サービスを再起動する方法は、「[httpd サービスの管理](#)」を参照してください。

設定に誤りがないことを確認するには、シェルプロンプトで以下のコマンドを実行します。

```
# apachectl configtest
Syntax OK
```

間違いからの復元を容易にするため、編集する前にオリジナルファイルのコピーを作成します。

1.4. HTTPD サービスの管理

本セクションでは、**httpd** サービスを起動、停止、および再起動する方法を説明します。

前提条件

- Apache HTTP Server がインストールされている。

手順

- **httpd** サービスを起動するには、以下を入力します。

```
# systemctl start httpd
```

- **httpd** サービスを停止するには、以下を入力します。

```
# systemctl stop httpd
```

- **httpd** サービスを再起動するには、次のコマンドを実行します。

```
# systemctl restart httpd
```

1.5. シングルインスタンスの APACHE HTTP サーバーの設定

本セクションでは、静的 HTML コンテンツを提供するために1つのインスタンスの Apache HTTP Server を設定する方法を説明します。

Web サーバーが、サーバーに関連付けられたすべてのドメインに対して同じコンテンツを提供する必要がある場合は、このセクションの手順に従います。異なるドメインに異なるコンテンツを提供する場合は、名前ベースの仮想ホストを設定します。詳細は「[Apache 名ベースの仮想ホストの設定](#)」を参照してください。

手順

1. **httpd** パッケージをインストールします。

```
# dnf install httpd
```

2. ローカルのファイアウォールで TCP ポート **80** を開きます。

```
# firewall-cmd --permanent --add-port=80/tcp
# firewall-cmd --reload
```

3. **httpd** サービスを有効にして起動します。

```
# systemctl enable --now httpd
```

4. オプション:HTML ファイルを `/var/www/html/` ディレクトリーに追加します。



注記

`/var/www/html/` にコンテンツを追加する場合は、デフォルトで **httpd** を実行するユーザーがファイルとディレクトリーを読み取る必要があります。コンテンツの所有者は、**root** ユーザーおよび **グループ**、または管理者別のユーザーまたはグループのいずれかになります。コンテンツの所有者が **root** ユーザーおよび **root** ユーザーグループである場合は、他のユーザーがファイルを読み取る必要があります。すべてのファイルとディレクトリーの SELinux コンテキストは、である必要があります。これはデフォルトで `/var/www` ディレクトリー内のすべてのコンテンツに適用されます。

検証手順

- Web ブラウザーで `http://server_IP_or_host_name/` に接続します。`/var/www/html/` ディレクトリーが空であるか、`index.html` または `index.htm` ファイルが含まれていない場合は、Apache が **Red Hat Enterprise Linux Test Page** を表示します。`/var/www/html/` に異なる名前の HTML ファイルが含まれる場合

は、`http://server_IP_or_host_name/example.html` のように、そのファイルに URL を入力して読み込むことができます。

関連情報

- Apache のマニュアルを参照してください。 [Installing the Apache HTTP Server manual](#) を参照してください。
- man ページの `httpd.service(8)` を参照してください。

1.6. APACHE 名ベースの仮想ホストの設定

名前ベースの仮想ホストは、Apache がサーバーの IP アドレスに解決する異なるドメインに対して異なるコンテンツを提供できるようにします。

本セクションの手順では、**example.com** ドメインおよび **example.net** ドメインの両方に、別のドキュメントルートディレクトリーを持つ仮想ホストを設定する方法を説明します。両方の仮想ホストが静的 HTML コンテンツに対応します。

前提条件

- クライアントおよび Web サーバーは、**example.com** および **example.net** ドメインを Web サーバーの IP アドレスに解決します。
これらのエントリーは DNS サーバーに手動で追加する必要がある点に注意してください。

手順

1. `httpd` パッケージをインストールします。

```
# dnf install httpd
```

2. `/etc/httpd/conf/httpd.conf` ファイルを編集します。
 - a. **example.com** ドメインの以下の仮想ホスト設定を追加します。

```
<VirtualHost *:80>
    DocumentRoot "/var/www/example.com/"
    ServerName example.com
    CustomLog /var/log/httpd/example.com_access.log combined
    ErrorLog /var/log/httpd/example.com_error.log
</VirtualHost>
```

これらの設定は以下を構成します。

- `<VirtualHost *:80>` ディレクティブのすべての設定は、この仮想ホストに固有のものです。
- **DocumentRoot** は、仮想ホストの Web コンテンツへのパスを設定します。
- **ServerName** は、この仮想ホストがコンテンツを提供するドメインを設定します。
複数のドメインを設定するには、**ServerAlias** パラメーターを設定に追加し、追加のドメインをこのパラメーターのスペースで区切ります。
- **CustomLog** は、仮想ホストのアクセスログへのパスを設定します。

- **ErrorLog** は、仮想ホストのエラーログへのパスを設定します。



注記

Apache は、**ServerName** パラメーターおよび **ServerAlias** パラメーターに設定したどのドメインにも一致しない要求にも、設定にある最初の仮想ホストを使用します。これには、サーバーの IP アドレスに送信される要求も含まれます。

3. **example.net** ドメイン向けに同様の仮想ホスト設定を追加します。

```
<VirtualHost *:80>
  DocumentRoot "/var/www/example.net/"
  ServerName example.net
  CustomLog /var/log/httpd/example.net_access.log combined
  ErrorLog /var/log/httpd/example.net_error.log
</VirtualHost>
```

4. 両方の仮想ホストのドキュメントルートを作成します。

```
# mkdir /var/www/example.com/
# mkdir /var/www/example.net/
```

5. **/var/www/**内にはない **DocumentRoot** パラメーターにパスを設定する場合は、両方のドキュメントルートに **httpd_sys_content_t** コンテキストを設定します。

```
# semanage fcontext -a -t httpd_sys_content_t "/srv/example.com(/.*)?"
# restorecon -Rv /srv/example.com/
# semanage fcontext -a -t httpd_sys_content_t "/srv/example.net(/.*)?"
# restorecon -Rv /srv/example.net/
```

以下のコマンドは、**/srv/example.com/** および **/srv/example.net/** ディレクトリーに **httpd_sys_content_t** コンテキストを設定します。

polycoreutils-python-utils パッケージをインストールして **restorecon** コマンドを実行する必要があります。

6. ローカルのファイアウォールで **80** ポートを開きます。

```
# firewall-cmd --permanent --add-port=80/tcp
# firewall-cmd --reload
```

7. **httpd** サービスを有効にして起動します。

```
# systemctl enable --now httpd
```

検証手順

1. 各仮想ホストのドキュメントルートに異なるサンプルファイルを作成します。

```
# echo "vHost example.com" > /var/www/example.com/index.html
# echo "vHost example.net" > /var/www/example.net/index.html
```

2. ブラウザーを使用して **http://example.com** に接続します。Web サーバーは、**example.com** 仮想ホストからのサンプルファイルを表示します。
3. ブラウザーを使用して **http://example.net** に接続します。Web サーバーは、**example.net** 仮想ホストからのサンプルファイルを表示します。

関連情報

- Apache 仮想ホストの設定に関する詳細は、Apache マニュアルの **仮想ホスト** のドキュメントを参照してください。マニュアルのインストールの詳細は、[「Apache HTTP Server のマニュアルのインストール」](#) を参照してください。

1.7. APACHE HTTP WEB サーバーの KERBEROS 認証の設定

Apache HTTP Web サーバーで Kerberos 認証を実行するために、RHEL 9 は **mod_auth_gssapi** Apache モジュールを使用します。Generic Security Services API (**GSSAPI**) は、Kerberos などのセキュリティライブラリーを使用する要求を行うアプリケーションのインターフェースです。**gssproxy** サービスでは、**httpd** サーバーに特権の分離を実装できます。これにより、セキュリティの観点からこのプロセスが最適化されます。



注記

削除した **mod_auth_kerb** モジュールは、**mod_auth_gssapi** モジュールに置き換わりません。

前提条件

- **httpd**、**mod_auth_gssapi**、および **gssproxy** パッケージがインストールされている。
- Apache Web サーバーが設定され、**httpd** サービスが実行している。

1.7.1. IdM 環境で GSS-Proxy の設定

この手順では、Apache HTTP Web サーバーで Kerberos 認証を実行するように **GSS-Proxy** を設定する方法を説明します。

手順

1. サービスプリンシパルを作成し、HTTP/<SERVER_NAME>@realm プリンシパルの **keytab** ファイルへのアクセスを有効にします。

```
# ipa service-add HTTP/<SERVER_NAME>
```

2. **/etc/gssproxy/http.keytab** ファイルに保存されているプリンシパルの **keytab** を取得します。

```
# ipa-getkeytab -s $(awk '/^server =/ {print $3}' /etc/ipa/default.conf) -k /etc/gssproxy/http.keytab -p HTTP/$(hostname -f)
```

このステップでは、パーミッションを 400 に設定すると、**root** ユーザーのみが **keytab** ファイルにアクセスできます。**apache** ユーザーは異なります。

3. 以下の内容で **/etc/gssproxy/80-httpd.conf** ファイルを作成します。

```
[service/HTTP]
```

```
mechs = krb5
cred_store = keytab:/etc/gssproxy/http.keytab
cred_store = ccache:/var/lib/gssproxy/clients/krb5cc_%U
euid = apache
```

4. **gssproxy** サービスを再起動して、有効にします。

```
# systemctl restart gssproxy.service
# systemctl enable gssproxy.service
```

関連情報

- **GSS-Proxy** の使用または調整の詳細は、**gssproxy(8)**、**gssproxy-mech(8)**、および **gssproxy.conf(5)** の man ページを参照してください。

1.7.2. Apache HTTP Web サーバーが共有するディレクトリーに Kerberos 認証の設定

この手順では、**/var/www/html/private/** ディレクトリーに Kerberos 認証を設定する方法を説明します。

前提条件

- **gssproxy** サービスが設定され、実行されている。

手順

1. **/var/www/html/private/** ディレクトリーを保護するように **mod_auth_gssapi** を設定します。

```
<Location /var/www/html/private>
  AuthType GSSAPI
  AuthName "GSSAPI Login"
  Require valid-user
</Location>
```

2. 以下の内容で **/etc/systemd/system/httpd.service** ファイルを作成します。

```
.include /lib/systemd/system/httpd.service
[Service]
Environment=GSS_USE_PROXY=1
```

3. **systemd** 設定をリロードします。

```
# systemctl daemon-reload
```

4. **httpd** サービスを再起動します。

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

検証手順

1. Kerberos チケットを取得します。

```
# kinit
```

2. ブラウザーで、保護されているディレクトリーの URL を開きます。

1.8. APACHE HTTP サーバーで TLS 暗号化の設定

デフォルトでは、Apache は暗号化されていない HTTP 接続を使用してクライアントにコンテンツを提供します。本セクションでは、TLS 暗号化を有効にし、Apache HTTP Server で頻繁に使用される暗号化関連の設定を行う方法を説明します。

前提条件

- Apache HTTP Server がインストールされ、実行している。

1.8.1. Apache HTTP Server への TLS 暗号化の追加

本セクションでは、**example.com** ドメインの Apache HTTP Server で TLS 暗号化を有効にする方法を説明します。

前提条件

- Apache HTTP Server がインストールされ、実行している。
- 秘密鍵は `/etc/pki/tls/private/example.com.key` ファイルに保存されます。秘密鍵および証明書署名要求 (CSR) を作成する方法と、認証局 (CA) からの証明書を要求する方法は、CA のドキュメントを参照してください。または、お使いの CA が ACME プロトコルに対応している場合は、`mod_md` モジュールを使用して、TLS 証明書の取得およびプロビジョニングを自動化できます。
- TLS 証明書は `/etc/pki/tls/certs/example.com.crt` ファイルに保存されます。別のパスを使用する場合は、この手順で対応する手順を調整します。
- 認証局証明書は `/etc/pki/tls/certs/ca.crt` に保存されています。別のパスを使用する場合は、この手順で対応する手順を調整します。
- クライアントおよび Web サーバーは、サーバーのホスト名を Web サーバーのホスト名に解決します。

手順

1. `mod_ssl` パッケージをインストールします。

```
# dnf install mod_ssl
```

2. `/etc/httpd/conf.d/ssl.conf` ファイルを編集し、以下の設定を `<VirtualHost _default_:443>` ディレクティブに追加します。
 - a. サーバー名を設定します。

```
ServerName example.com
```




重要

サーバー名は、証明書の **Common Name** フィールドに設定されているエントリーと一致している必要があります。

- b. オプション:**Subject Alt Names** (SAN) フィールドに追加のホスト名が含まれる場合には、これらのホスト名にも TLS 暗号化を提供するように **mod_ssl** を設定できます。これを設定するには、**ServerAliases** パラメーターと対応する名前を追加します。

```
ServerAlias www.example.com server.example.com
```

- c. 秘密鍵、サーバー証明書、および CA 証明書へのパスを設定します。

```
SSLCertificateKeyFile "/etc/pki/tls/private/example.com.key"
SSLCertificateFile "/etc/pki/tls/certs/example.com.crt"
SSLCACertificateFile "/etc/pki/tls/certs/ca.crt"
```

3. セキュリティー上の理由から、**root** ユーザーのみが秘密鍵ファイルにアクセスできるように設定します。

```
# chown root:root /etc/pki/tls/private/example.com.key
# chmod 600 /etc/pki/tls/private/example.com.key
```



警告

秘密鍵に権限のないユーザーがアクセスした場合は、証明書を取り消し、新しい秘密鍵を作成し、新しい証明書を要求します。そうでない場合は、TLS 接続が安全ではなくなります。

4. ローカルのファイアウォールでポート **443** を開きます。

```
# firewall-cmd --permanent --add-port=443/tcp
# firewall-cmd --reload
```

5. **httpd** サービスを再起動します。

```
# systemctl restart httpd
```



注記

パスワードで秘密鍵ファイルを保護した場合は、**httpd** サービスの起動時に毎回このパスワードを入力する必要があります。

検証手順

- ブラウザーを使用して、**https://example.com** に接続します。

関連情報

- Apache マニュアルの **SSL/TLS Encryption** のドキュメントを参照してください。
- [Installing the Apache HTTP Server manual](#) を参照してください。
- [RHEL 9 における TLS のセキュリティー上の検討事項](#)

1.8.2. Apache HTTP サーバーでサポートされる TLS プロトコルバージョンの設定

デフォルトでは、RHEL の Apache HTTP Server は、最新のブラウザにも互換性のある安全なデフォルト値を定義するシステム全体の暗号化ポリシーを使用します。たとえば、**DEFAULT** ポリシーは、apache で **TLSv1.2** プロトコルバージョンおよび **TLSv1.3** プロトコルバージョンのみが有効になっていることを定義します。

本セクションでは、Apache HTTP Server がサポートする TLS プロトコルバージョンを手動で設定する方法を説明します。たとえば、環境が特定の TLS プロトコルバージョンのみを有効にする必要がある場合には、以下の手順に従います。

- お使いの環境で、クライアントが弱い **TLS1** (TLSv1.0) プロトコルまたは **TLS1.1** プロトコルも使用できるようにする必要がある場合は、
- Apache が **TLSv1.2** プロトコルまたは **TLSv1.3** プロトコルのみに対応するように設定する場合。

前提条件

- [Apache HTTP Server への TLS 暗号化の追加](#) の説明に従って、TLS 暗号化がサーバーで有効になります。

手順

1. `/etc/httpd/conf/httpd.conf` ファイルを編集し、TLS プロトコルバージョンを設定する `<VirtualHost>` ディレクティブに以下の設定を追加します。たとえば、**TLSv1.3** プロトコルのみを有効にするには、以下を実行します。

```
SSLProtocol -All TLSv1.3
```

2. **httpd** サービスを再起動します。

```
# systemctl restart httpd
```

検証手順

1. 以下のコマンドを使用して、サーバーが **TLSv1.3** に対応していることを確認します。

```
# openssl s_client -connect example.com:443 -tls1_3
```

2. 以下のコマンドを使用して、サーバーが **TLSv1.2** に対応していないことを確認します。

```
# openssl s_client -connect example.com:443 -tls1_2
```

サーバーがプロトコルに対応していない場合、このコマンドは以下のエラーを返します。

```
140111600609088:error:1409442E:SSL routines:ssl3_read_bytes:tlsv1 alert protocol
version:ssl/record/rec_layer_s3.c:1543:SSL alert number 70
```

3. オプション:他の TLS プロトコルバージョンのコマンドを繰り返し実行します。

関連情報

- man ページの **update-crypto-policies(8)** を参照してください。
- [Using system-wide cryptographic policies](#) を参照してください。
- **SSLProtocol** パラメーターの詳細は、Apache マニュアルの **mod_ssl** ドキュメントを参照してください。
- [Installing the Apache HTTP Server manual](#) を参照してください。

1.8.3. Apache HTTP サーバーで対応している暗号の設定

デフォルトでは、Apache HTTP サーバーは、安全なデフォルト値を定義するシステム全体の暗号化ポリシーを使用します。これは、最近のブラウザとも互換性があります。システム全体の暗号化が許可する暗号化の一覧は、`/etc/crypto-policies/back-ends/openssl.config` ファイルを参照してください。

本セクションでは、Apache HTTP Server がサポートする暗号化を手動で設定する方法を説明します。お使いの環境で特定の暗号が必要な場合は、手順に従います。

前提条件

- [Apache HTTP Server への TLS 暗号化の追加](#) の説明に従って、TLS 暗号化がサーバーで有効になります。

手順

1. `/etc/httpd/conf/httpd.conf` ファイルを編集し、**SSLCipherSuite** パラメーターを、TLS 暗号を設定する `<VirtualHost>` ディレクティブに追加します。

```
SSLCipherSuite
"EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH:!SHA1:!SHA256"
```

この例では、**EECDH+AESGCM**、**EDH+AESGCM**、**AES256+EECDH**、および **AES256+EDH** 暗号のみを有効にし、**SHA1** および **SHA256** メッセージ認証コード (MAC) を使用するすべての暗号を無効にします。

2. **httpd** サービスを再起動します。

```
# systemctl restart httpd
```

検証手順

1. Apache HTTP Server が対応する暗号化のリストを表示するには、以下を行います。
 - a. **nmap** パッケージをインストールします。

```
# dnf install nmap
```

- b. **nmap** ユーティリティーを使用して、対応している暗号を表示します。

```
# nmap --script ssl-enum-ciphers -p 443 example.com
...
PORT      STATE SERVICE
443/tcp   open  https
| ssl-enum-ciphers:
|   TLSv1.2:
|     ciphers:
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (ecdh_x25519) - A
|       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (dh 2048) - A
|       TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (ecdh_x25519) - A
...
```

関連情報

- man ページの **update-crypto-policies(8)** を参照してください。
- [Using system-wide cryptographic policies](#) を参照してください。
- **SSLCipherSuite** パラメーターの詳細は、Apache マニュアルの **mod_ssl** ドキュメントを参照してください。
- [Installing the Apache HTTP Server manual](#) を参照してください。

1.9. TLS クライアント証明書認証の設定

クライアント証明書認証により、管理者は、証明書を使用して認証したユーザーのみが Web サーバーのリソースにアクセスできます。本セクションでは、`/var/www/html/Example/` ディレクトリーにクライアント証明書認証を設定する方法を説明します。

Apache HTTP Server が TLS 1.3 プロトコルを使用する場合、特定のクライアントには追加の設定が必要です。たとえば、Firefox で、**about:config** メニューの **security.tls.enable_post_handshake_auth** パラメーターを **true** に設定します。詳細は、「[Transport Layer Security version 1.3 in Red Hat Enterprise Linux 8](#)」を参照してください。

前提条件

- [Apache HTTP Server への TLS 暗号化の追加](#) の説明に従って、TLS 暗号化がサーバーで有効になります。

手順

1. `/etc/httpd/conf/httpd.conf` ファイルを編集し、以下の設定をクライアント認証を設定する `<VirtualHost>` ディレクティブに追加します。

```
<Directory "/var/www/html/Example/">
  SSLVerifyClient require
</Directory>
```

SSLVerifyClient require の設定では、`/var/www/html/Example/` ディレクトリーのコンテンツにクライアントがアクセスする前に、サーバーがクライアント証明書を正常に検証する必要があります。あることを定義します。

2. **httpd** サービスを再起動します。

```
# systemctl restart httpd
```

検証手順

1. **curl** ユーティリティを使用して、クライアント認証なしで **https://example.com/Example/** URL にアクセスします。

```
$ curl https://example.com/Example/  
curl: (56) OpenSSL SSL_read: error:1409445C:SSL routines:ssl3_read_bytes:tlsv13 **alert  
certificate required**, errno 0
```

このエラーは、Web サーバーにクライアント証明書認証が必要であることを示しています。

2. クライアントの秘密鍵と証明書、および CA 証明書を **curl** に渡して、クライアント認証で同じ URL にアクセスします。

```
$ curl --cacert ca.crt --key client.key --cert client.crt https://example.com/Example/
```

要求に成功すると、**curl** は **/var/www/html/Example/** ディレクトリーに保存されている **index.html** ファイルを表示します。

関連情報

- Apache マニュアルの **mod_ssl Configuration How-To** ドキュメントを参照してください。
- [Installing the Apache HTTP Server manual](#) を参照してください。

1.10. APACHE HTTP SERVER のマニュアルのインストール

本セクションでは、Apache HTTP Server のマニュアルをインストールする方法を説明します。本書では、以下のような詳細なドキュメントを提供します。

- 設定パラメーターおよびディレクティブ
- パフォーマンスチューニング
- 認証設定
- モジュール
- コンテンツのキャッシュ
- セキュリティーに関するヒント
- TLS 暗号化の設定

マニュアルをインストールした後は、Web ブラウザーを使用して表示できます。

前提条件

- Apache HTTP Server がインストールされ、実行している。

手順

1. **httpd-manual** パッケージをインストールします。

```
# dnf install httpd-manual
```

2. オプション:必要に応じて、デフォルトでは、Apache HTTP Server に接続するすべてのクライアントはマニュアルを表示できます。**192.0.2.0/24** サブネットなど、特定の IP 範囲へのアクセスを制限するには、`/etc/httpd/conf.d/manual.conf` ファイルを編集し、**Require ip 192.0.2.0/24** 設定を `<Directory "/usr/share/httpd/manual">` ディレクティブに追加します。

```
<Directory "/usr/share/httpd/manual">
...
**Require ip 192.0.2.0/24**
...
</Directory>
```

3. **httpd** サービスを再起動します。

```
# systemctl restart httpd
```

検証手順

1. Apache HTTP Server のマニュアルを表示するには、Web ブラウザーで `http://host_name_or_IP_address/manual/` に接続します。

1.11. モジュールの使用

httpd サービスは、モジュールアプリケーションであるため、多数の **Dynamic Shared Objects (DSO)** と一緒に配布されます。これは、必要に応じてランタイム時に、動的に読み込まれたり、読み込みが破棄されたりします。このモジュールは、`/usr/lib64/httpd/modules/` ディレクトリーにあります。

1.11.1. モジュールの読み込み

特定の DSO モジュールを読み込むには、**LoadModule** ディレクティブを使用します。別のパッケージが提供するモジュールは、多くの場合、`/etc/httpd/conf.modules.d/` ディレクトリーに独自の設定ファイルがあることに注意してください。

mod_ssl DSO の読み込み

```
LoadModule ssl_module modules/mod_ssl.so
```

モジュールの読み込み後、Web サーバーを再起動して設定を再読み込みします。**httpd** サービスを再起動する方法は、「[httpd サービスの管理](#)」を参照してください。

1.11.2. モジュールの作成

新しい DSO モジュールを作成するには、**httpd-devel** パッケージがインストールされていることを確認します。これを行うには、**root** で以下のコマンドを実行します。

```
# dnf install httpd-devel
```

このパッケージには、モジュールをコンパイルするために必要なインクルードファイル、ヘッダーファイル、および **APache eXtenSion (apxs)** ユーティリティが含まれます。

作成したら、以下のコマンドでモジュールを構築できます。

```
# apxs -i -a -c module_name.c
```

ビルドが成功すると、**Apache HTTP Server** で配布されるその他のモジュールと同じ方法でモジュールを読み込むことができます。

1.12. APACHE WEB サーバー設定で秘密鍵と証明書を使用できるように NSS データベースからの証明書のエクスポート

RHEL 8 以降、Apache Web サーバーに **mod_nss** モジュールが提供されなくなります。Red Hat は **mod_ssl** モジュールの使用を推奨します。秘密鍵と証明書を Network Security Services (NSS) データベースに保存する場合は、以下の [手順に従って](#)、[Privacy Enhanced Mail \(PEM\) 形式の鍵および証明書を抽出](#) します。

1.13. 関連情報

- **httpd(8) - httpd** サービスの man ページで、コマンドラインオプションの一覧が記載されません。
- **httpd.service(8) - httpd.service** ユニットファイルの man ページです。サービスのカスタマイズおよび機能強化の方法を説明します。
- **httpd.conf (5) - httpd** 設定ファイルの構造と場所を説明する **httpd** 設定用の man ページです。
- **apachectl (8) - Apache HTTP Server** の制御インターフェースの man ページです。
- Apache HTTP サーバーで Kerberos 認証を設定する方法の詳細は、「[Using GSS-Proxy for Apache httpd operation](#)」を参照してください。Kerberos の使用は、Apache HTTP Server でクライアント承認を強制する代替方法です。
- [PKCS #11 で暗号化ハードウェアを使用するようにアプリケーションを設定](#)

第2章 NGINX の設定および構成

NGINX は、たとえば、次のように使用できる高パフォーマンスなモジュラーサーバーです。

- Web サーバー
- リバースプロキシ
- ロードバランサー

本セクションでは、このシナリオで NGINX を行う方法を説明します。

2.1. NGINX のインストールおよび準備

Red Hat は、アプリケーションストリームを使用して NGINX の異なるバージョンを提供します。本セクションでは、以下を行う方法を説明します。

- ストリームを選択し、NGINX をインストールします。
- ファイアウォールで必要なポートを開きます。
- **nginx** サービスの有効化および開始

デフォルト設定を使用すると、NGINX はポート **80** の Web サーバーとして実行され、**/usr/share/nginx/html/** ディレクトリーのコンテンツを提供します。

前提条件

- RHEL 9 がインストールされている。
- ホストは Red Hat カスタマーポータルにサブスクライブしている。
- デフォルトでは、**firewalld** サービスは有効で開始しています。

手順

1. **nginx** パッケージをインストールします。

```
# dnf install nginx
```

2. NGINX がファイアウォールでサービスを提供するポートを開きます。たとえば、**firewalld** で HTTP (ポート 80) および HTTPS (ポート 443) のデフォルトポートを開くには、次のコマンドを実行します。

```
# firewall-cmd --permanent --add-port={80/tcp,443/tcp}
# firewall-cmd --reload
```

3. **nginx** サービスがシステムの起動時に自動的に起動するようにします。

```
# systemctl enable nginx
```

4. 必要に応じて、**nginx** サービスを起動します。

```
# systemctl start nginx
```


デフォルト設定を使用しない場合は、この手順を省略し、サービスを起動する前に NGINX を適切に設定します。

検証手順

1. **dnf** ユーティリティーを使用して、**nginx** パッケージがインストールされていることを確認します。

```
# dnf list installed nginx
Installed Packages
nginx.x86_64 1:1.20.1-4.el9 @rhel-AppStream
```

2. NGINX がサービスを提供するポートが **firewalld** で開いていることを確認します。

```
# firewall-cmd --list-ports
80/tcp 443/tcp
```

3. **nginx** サービスが有効になっていることを確認します。

```
# systemctl is-enabled nginx
enabled
```

関連情報

- Subscription Manager の詳細は、『[Subscription Manager の使用および設定](#)』ガイドを参照してください。
- ファイアウォールの設定に関する詳細は、『[ネットワークのセキュリティ保護](#)』を参照してください。

2.2. 異なるドメインに異なるコンテンツを提供する WEB サーバーとしての NGINX の設定

デフォルトでは、NGINX はサーバーの IP アドレスに関連付けられたすべてのドメイン名のクライアントに同じコンテンツを提供する Web サーバーとして動作します。この手順では、NGINX を設定する方法を説明します。

- **/var/www/example.com/** ディレクトリーのコンテンツで、**example.com** ドメインにリクエストを提供する。
- **/var/www/example.net/** ディレクトリーのコンテンツを使用して、**example.net** ドメインにリクエストを提供する。
- その他のすべてのリクエスト (たとえば、サーバーの IP アドレスまたはサーバーの IP アドレスに関連付けられたその他のドメイン) に **/usr/share/nginx/html/** ディレクトリーのコンテンツを指定します。

前提条件

- NGINX は、『[NGINX のインストールおよび準備](#)』の説明に従ってインストールされます。
- クライアントおよび Web サーバーは、**example.com** および **example.net** ドメインを Web サーバーの IP アドレスに解決します。

これらのエントリーは DNS サーバーに手動で追加する必要がある点に注意してください。

手順

1. `/etc/nginx/nginx.conf` ファイルを編集します。

- a. デフォルトでは、`/etc/nginx/nginx.conf` ファイルには `catch-all` 設定がすでに含まれています。設定からこの部分を削除した場合は、以下の **サーバー** ブロックを `/etc/nginx/nginx.conf` ファイルの **http** ブロックに再追加します。

```
server {
    listen      80 default_server;
    listen     [::]:80 default_server;
    server_name _;
    root       /usr/share/nginx/html;
}
```

これらの設定は以下を構成します。

- **listen** ディレクティブは、サービスがリッスンする IP アドレスとポートを定義します。この場合、NGINX は IPv4 と IPv6 の両方のアドレスのポート **80** でリッスンします。**default_server** パラメーターは、NGINX がこの **server** ブロックを IP アドレスとポートに一致するリクエストのデフォルトとして使用していることを示します。
- **server_name** パラメーターは、この **server** ブロックに対応するホスト名を定義します。**server_name** を `_` に設定すると、この **server** ブロックのホスト名を受け入れるように NGINX を設定します。
- **root** ディレクティブは、この **server** ブロックの Web コンテンツへのパスを設定します。

- b. **example.com** ドメインの同様の **server** ブロックを **http** ブロックに追加します。

```
server {
    server_name example.com;
    root       /var/www/example.com/;
    access_log /var/log/nginx/example.com/access.log;
    error_log  /var/log/nginx/example.com/error.log;
}
```

- **access_log** ディレクティブは、このドメインに別のアクセスログファイルを定義します。
- **error_log** ディレクティブは、このドメインに別のエラーログファイルを定義します。

- c. **example.net** ドメインの同様の **server** ブロックを **http** ブロックに追加します。

```
server {
    server_name example.net;
    root       /var/www/example.net/;
    access_log /var/log/nginx/example.net/access.log;
    error_log  /var/log/nginx/example.net/error.log;
}
```

2. 両方のドメインのルートディレクトリを作成します。

■

```
# mkdir -p /var/www/example.com/
# mkdir -p /var/www/example.net/
```

- 両方のルートディレクトリーに `httpd_sys_content_t` コンテキストを設定します。

```
# semanage fcontext -a -t httpd_sys_content_t "/var/www/example.com(/.*)?"
# restorecon -Rv /var/www/example.com/
# semanage fcontext -a -t httpd_sys_content_t "/var/www/example.net(/.*)?"
# restorecon -Rv /var/www/example.net/
```

これらのコマンドは、`/var/www/example.com/` ディレクトリーおよび `/var/www/example.net/` ディレクトリーに `httpd_sys_content_t` コンテキストを設定します。

`polycoreutils-python-utils` パッケージをインストールして `restorecon` コマンドを実行する必要があります。

- 両方のドメインのログディレクトリーを作成します。

```
# mkdir /var/log/nginx/example.com/
# mkdir /var/log/nginx/example.net/
```

- `nginx` サービスを再起動します。

```
# systemctl restart nginx
```

検証手順

- 各仮想ホストのドキュメントルートに異なるサンプルファイルを作成します。

```
# echo "Content for example.com" > /var/www/example.com/index.html
# echo "Content for example.net" > /var/www/example.net/index.html
# echo "Catch All content" > /usr/share/nginx/html/index.html
```

- ブラウザを使用して `http://example.com` に接続します。Web サーバーは、`/var/www/example.com/index.html` ファイルからのサンプルコンテンツを表示します。
- ブラウザを使用して `http://example.net` に接続します。Web サーバーは、`/var/www/example.net/index.html` ファイルからのサンプルコンテンツを表示します。
- ブラウザを使用して `http://IP_address_of_the_server` に接続します。Web サーバーは、`/usr/share/nginx/html/index.html` ファイルからのサンプルコンテンツを表示します。

2.3. NGINX WEB サーバーへの TLS 暗号化の追加

本セクションでは、`example.com` ドメインの NGINX Web サーバーで TLS 暗号化を有効にする方法を説明します。

前提条件

- NGINX は、「[NGINX のインストールおよび準備](#)」の説明に従ってインストールされます。
- 秘密鍵は `/etc/pki/tls/private/example.com.key` ファイルに保存されます。

秘密鍵および証明書署名要求 (CSR) を作成する方法と、認証局 (CA) からの証明書を要求する方法は、CA のドキュメントを参照してください。

- TLS 証明書は `/etc/pki/tls/certs/example.com.crt` ファイルに保存されます。別のパスを使用する場合は、この手順で対応する手順を調整します。
- CA 証明書がサーバーの TLS 証明書ファイルに追加されている。
- クライアントおよび Web サーバーは、サーバーのホスト名を Web サーバーのホスト名に解決します。
- ポート **443** が、ローカルのファイアウォールで開いている。

手順

1. `/etc/nginx/nginx.conf` ファイルを編集し、設定の **http** ブロックに以下の **server** ブロックを追加します。

```
server {
    listen      443 ssl;
    server_name example.com;
    root        /usr/share/nginx/html;
    ssl_certificate /etc/pki/tls/certs/example.com.crt;
    ssl_certificate_key /etc/pki/tls/private/example.com.key;
}
```

2. セキュリティ上の理由から、**root** ユーザーのみが秘密鍵ファイルにアクセスできるように設定します。

```
# chown root:root /etc/pki/tls/private/example.com.key
# chmod 600 /etc/pki/tls/private/example.com.key
```



警告

秘密鍵に権限のないユーザーがアクセスした場合は、証明書を取り消し、新しい秘密鍵を作成し、新しい証明書を要求します。そうでない場合は、TLS 接続が安全ではなくなります。

3. **nginx** サービスを再起動します。

```
# systemctl restart nginx
```

検証手順

- ブラウザーを使用して、`https://example.com` に接続します。

関連情報

- [RHEL 9 における TLS のセキュリティ上の検討事項](#)

2.4. HTTP トラフィックのリバースプロキシとしての NGINX の設定

NGINX Web サーバーは、HTTP トラフィックのリバースプロキシとして機能するように設定できます。たとえば、この機能を使用すると、リモートサーバーの特定のサブディレクトリーに要求を転送できます。クライアント側からは、クライアントはアクセスするホストからコンテンツを読み込みます。ただし、NGINX は実際のコンテンツをリモートサーバーから読み込み、クライアントに転送します。

この手順では、Web サーバーの `/example` ディレクトリーへのトラフィックを、URL `https://example.com` に転送する方法を説明します。

前提条件

- NGINX は、「[NGINX のインストールおよび準備](#)」の説明に従ってインストールされます。
- オプション:TLS 暗号化がリバースプロキシで有効になっている。

手順

1. `/etc/nginx/nginx.conf` ファイルを編集し、リバースプロキシを提供する `server` ブロックに以下の設定を追加します。

```
location /example {
    proxy_pass https://example.com;
}
```

`location` ブロックは、NGINX が `/example` ディレクトリーのすべての要求を `https://example.com` に渡すことを定義します。

2. SELinux プール値パラメーター `httpd_can_network_connect` を `1` に設定して、SELinux が NGINX がトラフィックを転送できるように設定します。

```
# setsebool -P httpd_can_network_connect 1
```

3. `nginx` サービスを再起動します。

```
# systemctl restart nginx
```

検証手順

- ブラウザーを使用して `http://host_name/example` に接続すると、`https://example.com` の内容が表示されます。

2.5. NGINX を HTTP ロードバランサーとして設定

NGINX リバースプロキシ機能を使用してトラフィックを負荷分散できます。この手順では、アクティブな接続数が最も少ない数に基づいて、異なるサーバーにリクエストを送信する HTTP ロードバランサーとして NGINX を設定する方法を説明します。両方のサーバーが利用できない場合、この手順はフォールバックの理由として 3 番目のホストも定義します。

前提条件

- NGINX は、「[NGINX のインストールおよび準備](#)」の説明に従ってインストールされます。

手順

1. `/etc/nginx/nginx.conf` ファイルを編集し、以下の設定を追加します。

```
http {
    upstream backend {
        least_conn;
        server server1.example.com;
        server server2.example.com;
        server server3.example.com backup;
    }

    server {
        location / {
            proxy_pass http://backend;
        }
    }
}
```

backend という名前のホストグループの **least_conn** ディレクティブは、アクティブな接続数が最も少ないホストによって NGINX が **server1.example.com** または **server2.example.com** に要求を送信することを定義します。NGINX は、他の 2 つのホストが利用できない場合は、**server3.example.com** のみをバックアップとして使用します。

proxy_pass ディレクティブを **http://backend** に設定すると、NGINX はリバースプロキシとして機能し、**backend** ホストグループを使用して、このグループの設定に基づいて要求を配信します。

least_conn 負荷分散メソッドの代わりに、以下を指定することができます。

- ラウンドロビンを使用し、サーバー全体で要求を均等に分散する方法はありません。
 - **ip_hash** - クライアントの、IPv4 アドレスの最初の 3 つのオクテット、または IPv6 アドレス全体から計算されたハッシュに基づいて、あるクライアントアドレスから同じサーバーに要求を送信します。
 - ユーザー定義のキーに基づいてサーバーを判断する **hash**。これは文字列、変数、または両方の組み合わせになります。**consistent** パラメーターは、ユーザー定義のハッシュ化された鍵の値に基づいて、NGINX がすべてのサーバーに要求を分散するように設定します。
 - **random**: 無作為に選択されたサーバーに要求を送信します。
2. **nginx** サービスを再起動します。

```
# systemctl restart nginx
```

2.6. 関連情報

- 公式の NGINX は、<https://nginx.org/en/docs/> を参照してください。Red Hat は本書を維持しておらず、インストールした NGINX バージョンでは機能しない可能性があることに注意してください。
- [PKCS #11 で暗号化ハードウェアを使用するようにアプリケーションを設定](#)

第3章 SQUID キャッシングプロキシサーバーの設定

Squid は、コンテンツをキャッシュして帯域幅を削減し、Web ページをより迅速に読み込むプロキシサーバーです。本章では、HTTP、HTTPS、FTP のプロトコルのプロキシとして Squid を設定する方法と、アクセスの認証および制限を説明します。

3.1. 認証なしで SQUID をキャッシングプロキシとして設定

本セクションでは、認証なしでキャッシュプロキシとして Squid の基本設定を説明します。以下の手順では、IP 範囲に基づいてプロキシへのアクセスを制限します。

前提条件

- `/etc/squid/squid.conf` ファイルが、**squid** パッケージにより提供されている。このファイルを編集した場合は、ファイルを削除して、パッケージを再インストールしている。

手順

1. **squid** パッケージをインストールします。

```
# dnf install squid
```

2. `/etc/squid/squid.conf` ファイルを編集します。

- a. **localnet** アクセス制御リスト (ACL) を、プロキシを使用できる IP 範囲と一致するように変更します。

```
acl localnet src 192.0.2.0/24
acl localnet 2001:db8:1::/64
```

デフォルトでは、`/etc/squid/squid.conf` ファイルには **localnet** ACL で指定されたすべての IP 範囲のプロキシを使用できるようにする **http_access allow localnet** ルールが含まれます。**http_access allow localnet** ルールの前に、**localnet** の ACL をすべて指定する必要があります。



重要

環境に一致しない既存の **acl localnet** エントリーをすべて削除します。

- b. 以下の ACL はデフォルト設定にあり、HTTPS プロトコルを使用するポートとして **443** を定義します。

```
acl SSL_ports port 443
```

ユーザーが他のポートでも HTTPS プロトコルを使用できるようにするには、ポートごとに ACL を追加します。

```
acl SSL_ports port port_number
```

- c. Squid が接続を確立できるポートに設定する **acl Safe_ports** ルールの一覧を更新します。たとえば、プロキシを使用するクライアントがポート 21 (FTP)、80 (HTTP)、443 (HTTPS) のリソースにのみアクセスできるようにするには、その設定の以下の **acl Safe_ports** ステートメントのみを保持します。

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

デフォルトでは、設定には、**Safe_ports** ACL に定義されていないポートへのアクセス拒否を定義する **http_access deny !Safe_ports** ルールが含まれています。

- d. **cache_dir** パラメーターにキャッシュの種類、キャッシュディレクトリーへのパス、キャッシュサイズ、さらにキャッシュの種類ごとの設定を構成します。

```
cache_dir ufs /var/spool/squid 10000 16 256
```

この設定により、以下が可能になります。

- Squid は、**ufs** キャッシュタイプを使用します。
 - Squid は、キャッシュを **/var/spool/squid/** ディレクトリーに保存します。
 - キャッシュのサイズが **10000** MB まで大きくなります。
 - Squid は、**16** 個のレベル1サブディレクトリーを **/var/spool/squid/** ディレクトリーに作成します。
 - Squid は、レベル1の各ディレクトリーに **256** 個のサブディレクトリーを作成します。**cache_dir** ディレクティブを設定しないと、Squid はキャッシュをメモリーに保存します。
3. **cache_dir** パラメーターに **/var/spool/squid/** 以外のキャッシュディレクトリーを設定する場合は、以下を行います。

- a. キャッシュディレクトリーを作成します。

```
# mkdir -p path_to_cache_directory
```

- b. キャッシュディレクトリーの権限を設定します。

```
# chown squid:squid path_to_cache_directory
```

- c. SELinux を、**Enforcing** モードで実行する場合は、キャッシュディレクトリー用に **squid_cache_t** コンテキストを設定します。

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

semanage ユーティリティーがシステムで利用できない場合は、**polycoreutils-python-utils** パッケージをインストールします。

4. ファイアウォールで **3128** ポートを開きます。

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

5. **squid** サービスを有効にして開始します。


```
# systemctl enable --now squid
```

検証手順

プロキシが正しく機能することを確認するには、**curl** ユーティリティを使用して Web ページをダウンロードします。

```
# curl -O -L "https://www.redhat.com/index.html" -x "proxy.example.com:3128"
```

curl でエラーが表示されず、**index.html** ファイルが現在のディレクトリーにダウンロードされている場合は、プロキシが動作します。

3.2. LDAP 認証を使用したキャッシングプロキシとしての SQUID の設定

本セクションでは、LDAP を使用してユーザーを認証するキャッシングプロキシとしての Squid の基本設定を説明します。この手順では、認証されたユーザーのみがプロキシを使用できるように設定します。

前提条件

- **/etc/squid/squid.conf** ファイルが、**squid** パッケージにより提供されている。このファイルを編集した場合は、ファイルを削除して、パッケージを再インストールしている。
- **uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com** などのサービスユーザーが LDAP ディレクトリーに存在します。Squid はこのアカウントを使用して認証ユーザーを検索します。認証ユーザーが存在する場合、Squid はこのユーザーをディレクトリーにバインドして、認証を確認します。

手順

1. **squid** パッケージをインストールします。

```
# dnf install squid
```

2. **/etc/squid/squid.conf** ファイルを編集します。

- a. **basic_ldap_auth** ヘルパーユーティリティを設定するには、**/etc/squid/squid.conf** に以下の設定エントリーを追加します。

```
auth_param basic program /usr/lib64/squid/basic_ldap_auth -b
"cn=users,cn=accounts,dc=example,dc=com" -D
"uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com" -W
/etc/squid/ldap_password -f "(&(objectClass=person)(uid=%s))" -ZZ -H
ldap://ldap_server.example.com:389
```

以下は、上記の例で **basic_ldap_auth** ヘルパーユーティリティに渡されるパラメーターについて説明しています。

- **-b base_DN** は LDAP 検索ベースを設定します。
- **-D proxy_service_user_DN** は、Squid が、ディレクトリー内の認証ユーザーを検索する際に使用するアカウントの識別名 (DN) を設定します。

- **-W path_to_password_file** は、プロキシサービスユーザーのパスワードが含まれるファイルへのパスを設定します。パスワードファイルを使用すると、オペレーティングシステムのプロセス一覧にパスワードが表示されなくなります。
 - **-f LDAP_filter** は、LDAP 検索フィルターを指定します。Squid は、**%s** 変数を、認証ユーザーが提供するユーザー名に置き換えます。
上記の例の **(&(objectClass=person)(uid=%s))** フィルターは、ユーザー名が **uid** 属性に設定された値と一致する必要があり、ディレクトリーエントリーに **person** オブジェクトクラスが含まれることを定義します。
 - **-ZZ** は、**STARTTLS** コマンドを使用して、LDAP プロトコルで TLS 暗号化接続を強制します。以下の状況で **-ZZ** を省略します。
 - LDAP サーバーは、暗号化された接続にを対応しません。
 - URL に指定されたポートは、LDAPS プロトコルを使用します。
 - **-H LDAP_URL** パラメーターは、プロトコル、ホスト名、IP アドレス、および LDAP サーバーのポートを URL 形式で指定します。
- b. 以下の ACL およびルールを追加して、Squid で、認証されたユーザーのみがプロキシを使用できるように設定します。

```
acl ldap-auth proxy_auth REQUIRED
http_access allow ldap-auth
```



重要

これらの設定は、**http_access deny all** ルールの前に指定します。

- c. 次のルールを削除して、**localnet** ACL で指定された IP 範囲のプロキシ認証の回避を無効にします。

```
http_access allow localnet
```

- d. 以下の ACL はデフォルト設定にあり、HTTPS プロトコルを使用するポートとして **443** を定義します。

```
acl SSL_ports port 443
```

ユーザーが他のポートでも HTTPS プロトコルを使用できるようにするには、ポートごとに ACL を追加します。

```
acl SSL_ports port port_number
```

- e. Squid が接続を確立できるポートに設定する **acl Safe_ports** ルールの一覧を更新します。たとえば、プロキシを使用するクライアントがポート 21 (FTP)、80 (HTTP)、443 (HTTPS) のリソースにのみアクセスできるようにするには、その設定の以下の **acl Safe_ports** ステートメントのみを保持します。

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

デフォルトでは、設定には、**Safe_ports ACL** に定義されていないポートにアクセス拒否を定義する **http_access deny !Safe_ports** ルールが含まれています。

- f. **cache_dir** パラメーターにキャッシュの種類、キャッシュディレクトリーへのパス、キャッシュサイズ、さらにキャッシュの種類ごとの設定を構成します。

```
cache_dir ufs /var/spool/squid 10000 16 256
```

この設定により、以下が可能になります。

- Squid は、**ufs** キャッシュタイプを使用します。
 - Squid は、キャッシュを **/var/spool/squid/** ディレクトリーに保存します。
 - キャッシュのサイズが **10000** MB まで大きくなります。
 - Squid は、**16** 個のレベル1サブディレクトリーを **/var/spool/squid/** ディレクトリーに作成します。
 - Squid は、レベル1の各ディレクトリーに **256** 個のサブディレクトリーを作成します。**cache_dir** ディレクティブを設定しないと、Squid はキャッシュをメモリーに保存します。
3. **cache_dir** パラメーターに **/var/spool/squid/** 以外のキャッシュディレクトリーを設定する場合は、以下を行います。

- a. キャッシュディレクトリーを作成します。

```
# mkdir -p path_to_cache_directory
```

- b. キャッシュディレクトリーの権限を設定します。

```
# chown squid:squid path_to_cache_directory
```

- c. SELinux を、**Enforcing** モードで実行する場合は、キャッシュディレクトリー用に **squid_cache_t** コンテキストを設定します。

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"
# restorecon -Rv path_to_cache_directory
```

semanage ユーティリティーがシステムで利用できない場合は、**policycoreutils-python-utils** パッケージをインストールします。

4. LDAP サービスユーザーのパスワードを **/etc/squid/ldap_password** ファイルに保存し、そのファイルに適切なパーミッションを設定します。

```
# echo "password" > /etc/squid/ldap_password
# chown root:squid /etc/squid/ldap_password
# chmod 640 /etc/squid/ldap_password
```

5. ファイアウォールで **3128** ポートを開きます。

```
# firewall-cmd --permanent --add-port=3128/tcp
# firewall-cmd --reload
```

6. **squid** サービスを有効にして開始します。

```
# systemctl enable --now squid
```

検証手順

プロキシが正しく機能することを確認するには、**curl** ユーティリティーを使用して Web ページをダウンロードします。

```
# curl -O -L "https://www.redhat.com/index.html" -x  
"user_name:password@proxy.example.com:3128"
```

curl でエラーが表示されず、**index.html** ファイルが現在のディレクトリーにダウンロードされている場合は、プロキシが動作します。

トラブルシューティングの手順

ヘルパーユーティリティーが正しく機能していることを確認するには、以下の手順を行います。

1. **auth_param** パラメーターで使ったのと同じ設定で、ヘルパーユーティリティーを手動で起動します。

```
# /usr/lib64/squid/basic_ldap_auth -b "cn=users,cn=accounts,dc=example,dc=com" -D  
"uid=proxy_user,cn=users,cn=accounts,dc=example,dc=com" -W  
/etc/squid/ldap_password -f "(&(objectClass=person)(uid=%s))" -ZZ -H  
ldap://ldap_server.example.com:389
```

2. 有効なユーザー名とパスワードを入力し、**Enter** を押します。

```
user_name password
```

ヘルパーユーティリティーが **OK** を返すと、認証に成功しました。

3.3. KERBEROS 認証を使用したキャッシングプロキシとしての SQUID の設定

本セクションでは、Kerberos を使用して Active Directory (AD) にユーザーを認証するキャッシングプロキシとしての Squid 基本設定を説明します。この手順では、認証されたユーザーのみがプロキシを使用できるように設定します。

前提条件

- **/etc/squid/squid.conf** ファイルが、**squid** パッケージにより提供されている。このファイルを編集した場合は、ファイルを削除して、パッケージを再インストールしている。
- Squid をインストールするサーバーが、AD ドメインのメンバーである。

手順

1. 以下のパッケージをインストールします。

```
dnf install squid krb5-workstation
```

2. AD ドメイン管理者として認証します。

```
# kinit administrator@AD.EXAMPLE.COM
```

- Squid 用のキータブを作成し、これを `/etc/squid/HTTP.keytab` ファイルに保存します。

```
# export KRB5_KTNAME=FILE:/etc/squid/HTTP.keytab
# net ads keytab CREATE -U administrator
```

- HTTP** サービスプリンシパルをキータブに追加します。

```
# net ads keytab ADD HTTP -U administrator
```

- キータブファイルの所有者を **squid** ユーザーに設定します。

```
# chown squid /etc/squid/HTTP.keytab
```

- 必要に応じて、キータブファイルに、プロキシサーバーの完全修飾ドメイン名 (FQDN) の **HTTP** サービスプリンシパルが含まれていることを確認します。

```
klist -k /etc/squid/HTTP.keytab
Keytab name: FILE:/etc/squid/HTTP.keytab
KVNO Principal
-----
...
 2 HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
...
```

- `/etc/squid/squid.conf` ファイルを編集します。

- negotiate_kerberos_auth** ヘルパーユーティリティーを設定するには、`/etc/squid/squid.conf` 部に以下の設定エントリーを追加します。

```
auth_param negotiate program /usr/lib64/squid/negotiate_kerberos_auth -k
/etc/squid/HTTP.keytab -s HTTP/proxy.ad.example.com@AD.EXAMPLE.COM
```

以下は、上記の例の **negotiate_kerberos_auth** ヘルパーユーティリティーに渡されるパラメーターを説明します。

- **-k ファイル** は、キータブファイルへのパスを設定します。squid ユーザーには、このファイルに対する読み取り権限があることに注意してください。
- **-s HTTP/host_name@kerberos_realm** は、Squid が使用する Kerberos プリンシパルを設定します。
必要に応じて、以下のパラメーターのいずれかまたは両方をヘルパーユーティリティーに渡すことによりログインを有効にできます。
- **-i** は、認証ユーザーなどの情報メッセージをログに記録します。
- **-d** は、デバッグロギングを有効にします。
Squid は、ヘルパーユーティリティーから、`/var/log/squid/cache.log` ファイルのログにデバッグ情報を記録します。

- 以下の ACL およびルールを追加して、Squid で、認証されたユーザーのみがプロキシを使用できるように設定します。

```
acl kerb-auth proxy_auth REQUIRED
http_access allow kerb-auth
```



重要

http_access deny all ルールの前にこの設定を指定します。

- c. 次のルールを削除して、**localnet** ACL で指定された IP 範囲のプロキシ認証の回避を無効にします。

```
http_access allow localnet
```

- d. 以下の ACL はデフォルト設定にあり、HTTPS プロトコルを使用するポートとして **443** を定義します。

```
acl SSL_ports port 443
```

ユーザーが他のポートでも HTTPS プロトコルを使用できるようにするには、ポートごとに ACL を追加します。

```
acl SSL_ports port port_number
```

- e. Squid が接続を確立できるポートに設定する **acl Safe_ports** ルールの一覧を更新します。たとえば、プロキシを使用するクライアントがポート 21 (FTP)、80 (HTTP)、443 (HTTPS) のリソースにのみアクセスできるようにするには、その設定の以下の **acl Safe_ports** ステートメントのみを保持します。

```
acl Safe_ports port 21
acl Safe_ports port 80
acl Safe_ports port 443
```

デフォルトでは、設定には、**Safe_ports** ACL に定義されていないポートへのアクセス拒否を定義する **http_access deny !Safe_ports** ルールが含まれています。

- f. **cache_dir** パラメーターにキャッシュの種類、キャッシュディレクトリーへのパス、キャッシュサイズ、さらにキャッシュの種類ごとの設定を構成します。

```
cache_dir ufs /var/spool/squid 10000 16 256
```

この設定により、以下が可能になります。

- Squid は、**ufs** キャッシュタイプを使用します。
- Squid は、キャッシュを **/var/spool/squid/** ディレクトリーに保存します。
- キャッシュのサイズが **10000** MB まで大きくなります。
- Squid は、**16** 個のレベル1サブディレクトリーを **/var/spool/squid/** ディレクトリーに作成します。
- Squid は、レベル1の各ディレクトリーに **256** 個のサブディレクトリーを作成します。**cache_dir** ディレクティブを設定しないと、Squid はキャッシュをメモリーに保存します。

8. `cache_dir` パラメーターに `/var/spool/squid/` 以外のキャッシュディレクトリーを設定する場合は、以下を行います。

- a. キャッシュディレクトリーを作成します。

```
# mkdir -p path_to_cache_directory
```

- b. キャッシュディレクトリーの権限を設定します。

```
# chown squid:squid path_to_cache_directory
```

- c. SELinux を、**Enforcing** モードで実行する場合は、キャッシュディレクトリー用に `squid_cache_t` コンテキストを設定します。

```
# semanage fcontext -a -t squid_cache_t "path_to_cache_directory(/.*)?"  
# restorecon -Rv path_to_cache_directory
```

semanage ユーティリティーがシステムで利用できない場合は、**polycoreutils-python-utils** パッケージをインストールします。

9. ファイアウォールで **3128** ポートを開きます。

```
# firewall-cmd --permanent --add-port=3128/tcp  
# firewall-cmd --reload
```

10. **squid** サービスを有効にして開始します。

```
# systemctl enable --now squid
```

検証手順

プロキシが正しく機能することを確認するには、**curl** ユーティリティーを使用して Web ページをダウンロードします。

```
# curl -O -L "https://www.redhat.com/index.html" --proxy-negotiate -u : -x  
"proxy.ad.example.com:3128"
```

curl でエラーが表示されず、**index.html** ファイルが現在のディレクトリーに存在する場合は、プロキシが機能します。

トラブルシューティングの手順

Kerberos 認証を手動でテストするには、以下を行います。

1. AD アカウントの Kerberos チケットを取得します。

```
# kinit user@AD.EXAMPLE.COM
```

2. 必要に応じて、キーを表示します。

```
# klist
```

3. **negotiate_kerberos_auth_test** ユーティリティーを使用して認証をテストします。

```
# /usr/lib64/squid/negotiate_kerberos_auth_test proxy.ad.example.com
```

ヘルパーユーティリティーがトークンを返す場合は、認証に成功しました。

```
Token: YlIFtAYGKwYBBQUColIFqDC...
```

3.4. SQUID でのドメイン拒否リストの設定

多くの場合、管理者は特定のドメインへのアクセスをブロックする必要があります。本セクションでは、Squid でドメインの拒否リストを設定する方法を説明します。

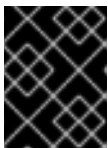
前提条件

- Squid が設定され、ユーザーはプロキシを使用できます。

手順

1. `/etc/squid/squid.conf` ファイルを編集し、以下の設定を追加します。

```
acl domain_deny_list dstdomain "/etc/squid/domain_deny_list.txt"
http_access deny all domain_deny_list
```



重要

ユーザーまたはクライアントへのアクセスを許可する最初の `http_access allow` ステートメントの前に、これらのエントリーを追加します。

2. `/etc/squid/domain_deny_list.txt` ファイルを作成し、ブロックするドメインを追加します。たとえば、サブドメインを含む `example.com` へのアクセスをブロックし、`example.net` をブロックするには、次の設定を追加します。

```
.example.com
example.net
```



重要

squid 設定の `/etc/squid/domain_deny_list.txt` ファイルを参照している場合は、このファイルは空にすることはできません。このファイルが空の場合、Squid は起動できません。

3. `squid` サービスを再起動します。

```
# systemctl restart squid
```

3.5. SQUID サービスが特定のポートまたは IP アドレスをリッスンするように設定

デフォルトでは、Squid プロキシサービスはすべてのネットワークインターフェースの `3128` ポートでリッスンします。本セクションでは、ポートを変更し、Squid が特定の IP アドレスをリッスンするように設定する方法を説明します。

前提条件

- **squid** パッケージがインストールされている。

手順

1. `/etc/squid/squid.conf` ファイルを編集します。

- Squid サービスがリッスンするポートを設定するには、**http_port** パラメーターにポート番号を設定します。たとえば、ポートを **8080** に設定するには、次を設定します。

```
http_port 8080
```

- Squid サービスがリッスンする IP アドレスを設定するには、**http_port** パラメーターに IP アドレスとポート番号を設定します。たとえば、Squid がポート **3128** の IP アドレス **192.0.2.1** でのみリッスンするように設定するには、次を設定します。

```
http_port 192.0.2.1:3128
```

複数の **http_port** パラメーターを設定ファイルに追加して、Squid が複数のポートおよび IP アドレスでリッスンするように設定します。

```
http_port 192.0.2.1:3128
http_port 192.0.2.1:8080
```

2. Squid が別のポートをデフォルト (**3128**) として使用するよう設定する場合は、以下を行います。
 - a. ファイアウォールのポートを開きます。

```
# firewall-cmd --permanent --add-port=port_number/tcp
# firewall-cmd --reload
```

- b. SELinux を Enforcing モードで実行する場合は、ポートを **squid_port_t** ポートタイプ定義に割り当てます。

```
# semanage port -a -t squid_port_t -p tcp port_number
```

semanage ユーティリティーがシステムで利用できない場合は、**policycoreutils-python-utils** パッケージをインストールします。

3. **squid** サービスを再起動します。

```
# systemctl restart squid
```

3.6. 関連情報

- `/etc/squid/squid.conf` で設定できる設定パラメーターの一覧と詳細な説明は、`usr/share/doc/squid-<version>/squid.conf.documented` ファイルを参照してください。

