SSL Linux

- Configure local https on ubuntu
- install OpenSSL locally on Ubuntu

Configure local https on ubuntu

T0 set up a local HTTPS server on Ubuntu, use a tool like `mkcert` for simplicity or `openssl` for more control, and then configure your web server (e.g., Apache) to use the generated certificates. The recommended method is to install and use `mkcert`, which generates a trusted, local Certificate Authority (CA) to create self-signed certificates for `localhost` or custom hostnames without browser warnings. Alternatively, you can generate a self-signed certificate using `openssl`, which requires more manual steps to create the key, certificate, and to trust it in your browser.

Method 1: Using `mkcert` (Recommended)

This video demonstrates how to set up HTTPS with mkcert and enable HTTPS for localhost:

Unconventional Coding

YouTube • 31 Dec 2022

1. **Install `mkcert`**: Install the tool to manage local certificate authorities and certificates.
   - `sudo apt install mkcert`
2. **Install the local CA**: Run `mkcert` to install a local CA that your system will trust.
   - `mkcert -install`
3. **Create a certificate**: Generate a certificate and key for your local site (e.g., `localhost`) or domain. Point to note here in case of domain, ip address should be mapped to it before this command.
   - `mkcert localhost 127.0.0.1`
   - `sudo mkcert drupalwithdata.offline.com`

Configure your server: Configure your specific application or web server to use the generated `localhost.pem` (certificate) and `localhost-key.pem` (key) files.

Method 2: Using openssl

1. **Install `openssl`**: Ensure `openssl` is installed. It is usually pre-installed on Ubuntu.
2. **Generate a private key and a certificate signing request (CSR)**:
   - `openssl req -new -newkey rsa:2048 -nodes -keyout localhost.key -out localhost.csr`

3. **Generate the self-signed certificate**: Use the CSR and private key to create the certificate.
   - `openssl x509 -req -days 365 -in localhost.csr -signkey localhost.key -out localhost.crt`
4. **Configure your web server**:
   - **For Apache**:
     - Enable the SSL module: `sudo a2enmod ssl`.
     - Configure your site's virtual host to point to your certificate (`localhost.crt`) and private key (`localhost.key`) files.
     - Restart Apache: `sudo systemctl restart apache2`.
   - **For other servers**: Configure your server's settings to use the `.crt` and `.key` files.
5. **Trust the certificate**:
   - This is an optional but recommended step to avoid browser warnings.
   - Install certificate utilities: `sudo apt-get install libnss3-tools`.
   - Import and trust the certificate into the browser's database (using `pk12util` for Firefox or `certutil` for other applications).

Final steps for both methods

- After generating the certificates and configuring your server, you may still see a browser warning because the certificate is "self-signed" and not issued by a public Certificate Authority.
- You will need to accept the security risk to proceed to your local serve

install OpenSSL locally on Ubuntu

To install OpenSSL locally on Ubuntu, you can compile and install it from source by first installing build tools, then downloading and extracting the OpenSSL source code, and finally running the config, make, and make install commands with the appropriate prefix and flags. For the simplest local installation, download the source, change to the directory, and use ./config --prefix=/openssl --openssldir=/openssl, followed by make and make install, then update your `~/.bash_profile` with the correct `PATH` and `LD_LIBRARY_PATH`.

Method 1: Install from source to a local directory

This method installs a specific version of OpenSSL to a custom directory, like `~/openssl`, so it doesn't interfere with the system-installed version.

bash

```
# Install build tools
sudo apt update
sudo apt install build-essential zlib1g-dev

# Create a working directory and download OpenSSL source
cd /usr/local/src/
wget https://www.openssl.org/source/openssl-1.1.1k.tar.gz
tar -xf openssl-1.1.1k.tar.gz
cd openssl-1.1.1k

# Configure and build (using --prefix to specify local install)
./config --prefix=~/openssl --openssldir=~/openssl
make

# Install to the local directory
make install
```

Method 2: Update your environment variables

After installing from source, you must tell your shell where to find the new binaries and libraries.

1. **Edit the bash profile**:
   bash
2. ```
   nano ~/.bash_profile
   ```
3. **Add the following lines at the end of the file** (adjusting `username` if necessary):
   bash
4. ```
   export PATH=$HOME/openssl/bin:$PATH
   export LD_LIBRARY_PATH=$HOME/openssl/lib:$LD_LIBRARY_PATH
   export LDFLAGS="-L $HOME/openssl/lib -Wl,-rpath,$HOME/openssl/lib"
   ```
5. **Save and close** the file, then reload your profile:
   bash

```
source ~/.bash_profile
```

Method 3: Verify the installation

1. **Check the version** to ensure you are using the new installation:
   bash
2. ```
   openssl version
   ```

   This should output the version you installed (e.g., `OpenSSL 1.1.1k 25 Mar 2025`).

Verify the path to confirm it's the local one:bash

```
which openssl
```

This should point to `~/openssl/bin/openssl`
```