.github	Drop "go vet" from CI due to false positives	last year
AUTHORS	LICENSE+AUTHORS: follow appropriate company policies	5 years ago
LICENSE	LICENSE+AUTHORS: follow appropriate company policies	5 years ago
README.md	Add dl.filippo.io instructions to the README	last year
cert.go	Correct failed to save CA certificate message (#399)	last year
go.mod	Update dependencies	last year
go.sum	Update dependencies	last year
main.go	Only print help when called without arguments	last year
truststore_darwin.go	Add support for Firefox in a Snap for Ubuntu 22.04	last year
truststore_java.go	truststore_nss: retry certtool with sudo when it fails due to p...	4 years ago
truststore_linux.go	Add support for Firefox in a Snap for Ubuntu 22.04	last year
truststore_nss.go	Add support for Firefox in a Snap for Ubuntu 22.04	last year
truststore_windows.go	Add support for Firefox in a Snap for Ubuntu 22.04	last year

README BSD-3-Clause license

mkcert

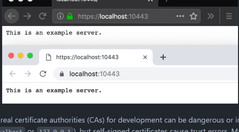
mkcert is a simple tool for making locally-trusted development certificates. It requires no configuration.

```
$ mkcert -install
Created a new local CA 💥
The local CA is now installed in the system trust store! ⚡
The local CA is now installed in the Firefox trust store (requires browser restart)! 🦊

$ mkcert example.com "*.example.com" example.test localhost 127.0.0.1 ::1

Created a new certificate valid for the following names 📜
 - "example.com"
 - "*.example.com"
 - "example.test"
 - "localhost"
 - "127.0.0.1"
 - "::1"

The certificate is at "./example.com+5.pem" and the key at "./example.com+5-key.pem" ✅
```



Using certificates from real certificate authorities (CAs) for development can be dangerous or impossible (for hosts like `example.test`, `localhost` or `127.0.0.1`), but self-signed certificates cause trust errors. Managing your own CA is the best solution, but usually involves arcane commands, specialized knowledge and manual steps.

mkcert automatically creates and installs a local CA in the system root store, and generates locally-trusted certificates. mkcert does not automatically configure servers to use the certificates, though, that's up to you.

Installation

> Warning: the `rootCA-key.pem` file that mkcert automatically generates gives complete power to intercept secure requests from your machine. Do not share it.

macOS

On macOS, use Homebrew

```
brew install mkcert
brew install nss # if you use Firefox
```

or MacPorts.

```
sudo port selfupdate
sudo port install mkcert
sudo port install nss # if you use Firefox
```

Linux

On Linux, first install `certutil`.

```
sudo apt install libnss3-tools
    -or-
sudo yum install nss-tools
    -or-
sudo pacman -S nss
    -or-
sudo zypper install mozilla-nss-tools
```

Then you can install using Homebrew on Linux

```
brew install mkcert
```

or build from source (requires Go 1.13+)

```
git clone https://github.com/FiloSottile/mkcert && cd mkcert
go build -ldflags "-X main.Version=$(git describe --tags)"
```

or use the pre-built binaries.

```
curl -JLO "https://dl.filippo.io/mkcert/latest?for=linux/amd64"
chmod +x mkcert-v*-linux-amd64
sudo cp mkcert-v*-linux-amd64 /usr/local/bin/mkcert
```

For Arch Linux users, `mkcert` is available on the official Arch Linux repository.

```
sudo pacman -Syu mkcert
```

Windows

On Windows, use Chocolatey

```
choco install mkcert
```

or use Scoop

```
scoop bucket add extras
scoop install mkcert
```

or build from source (requires Go 1.10+), or use the pre-built binaries.

If you're running into permission problems try running `mkcert` as an Administrator.

Supported root stores

mkcert supports the following root stores:

- macOS system store
- Windows system store
- Linux variants that provide either
  - `update-ca-trust` (Fedora, RHEL, CentOS) or
  - `update-ca-certificates` (Ubuntu, Debian, OpenSUSE, SLES) or
  - `trust` (Arch)
- Firefox (macOS and Linux only)
- Chrome and Chromium
- Java (when `JAVA_HOME` is set)

To only install the local root CA into a subset of them, you can set the `TRUST_STORES` environment variable to a comma-separated list. Options are: "system", "java" and "nss" (includes Firefox).

Advanced topics

Advanced options

```
    -cert-file FILE, -key-file FILE, -p12-file FILE
        Customize the output paths.

    -client
        Generate a certificate for client authentication.

    -ecdsa
        Generate a certificate with an ECDSA key.

    -pkcs12
        Generate a ".p12" PKCS #12 file, also know as a ".pfx" file,
        containing certificate and key for legacy applications.

    -csr CSR
        Generate a certificate based on the supplied CSR. Conflicts with
        all other flags and arguments except -install and -cert-file.
```

> Note: You *must* place these options before the domain names list.

Example

```
mkcert -key-file key.pem -cert-file cert.pem example.com *.example.com
```

S/MIME

mkcert automatically generates an S/MIME certificate if one of the supplied names is an email address.

```
mkcert filippo@example.com
```

Mobile devices

For the certificates to be trusted on mobile devices, you will have to install the root CA. It's the `rootCA.pem` file in the folder printed by `mkcert -CAROOT`.

On iOS, you can either use AirDrop, email the CA to yourself, or serve it from an HTTP server. After opening it, you need to install the profile in Settings > Profile Downloaded and then enable full trust in it.

For Android, you will have to install the CA and then enable user roots in the development build of your app. See this StackOverflow answer.

Using the root with Node.js

Node does not use the system root store, so it won't accept mkcert certificates automatically. Instead, you will have to set the `NODE_EXTRA_CA_CERTS` environment variable.

```
export NODE_EXTRA_CA_CERTS="$(mkcert -CAROOT)/rootCA.pem"
```

Changing the location of the CA files

The CA certificate and its key are stored in an application data folder in the user home. You usually don't have to worry about it, as installation is automated, but the location is printed by `mkcert -CAROOT`.

If you want to manage separate CAs, you can use the environment variable `$CAROOT` to set the folder where mkcert will place and look for the local CA files.

Installing the CA on other systems

Installing in the trust store does not require the CA key, so you can export the CA certificate and use mkcert to install it in other machines.

- Look for the `rootCA.pem` file in `mkcert -CAROOT`
- copy it to a different machine
- set `$CAROOT` to its directory
- run `mkcert -install`

Remember that mkcert is meant for development purposes, not production, so it should not be used on end users' machines, and that you should *not* export or share `rootCA-key.pem`.