mkcert Local trusted dev certs

START HERE

`mkcert -install`
Create CA, trust it

`mkcert -CAROOT`
Where the CA lives

`mkcert <name> ...`
Issue a certificate

`mkcert -cert-file <f> -key-file <f> <name>`
Choose output names

`mkcert localhost 127.0.0.1 ::1`
Local dev cert

`mkcert -pkcs12 <name>`
Also write .p12

`mkcert "*.<domain>"`
Wildcard cert

`mkcert -uninstall`
Remove CA from stores

CONCEPTS

mkcert makes a private CA on your machine, trusts it, and signs certificates with it. Browsers then accept your dev certs with no warning. Never share rootCA-key.pem: whoever has it can impersonate any site to you. Dev use only, never production.

mkcert -install CA in CAROOT
↓ trusts
Trust stores system, NSS, Java
↓ signs
Leaf cert + key your hostnames
One CA, many leaf certificates

Put flags before the names, otherwise they are ignored. mkcert does not configure your server: point it at the generated .pem and -key.pem files yourself.

SETUP

`mkcert -install`
Create local CA and trust it

`mkcert -uninstall`
Remove CA from trust stores
Does not delete the CA files; remove the CAROOT folder for that.

`mkcert -CAROOT`
Print CA folder

SETUP (cont.)

`sudo apt install libnss3-tools`
Linux: Firefox/Chrome NSS support
Needs certutil before -install, or browsers will not trust the CA.

`brew install mkcert nss`
macOS install (nss for Firefox)

`choco install mkcert`
Windows install

`TRUST_STORES=system,nss mkcert -install`
Limit which stores are used
Values: system, java, nss. Java also needs JAVA_HOME set.

ISSUE CERTIFICATES

`mkcert <name> ...`
Cert for one or more names
Output: <first-name>+<n>.pem and <first-name>+<n>-key.pem in the current directory.

`mkcert example.test localhost 127.0.0.1 ::1`
Hostnames plus IPs

`mkcert "*.<domain>"`
Wildcard certificate

`mkcert -cert-file <crt> -key-file <key> <name>`
Set output file names

`mkcert -ecdsa <name>`
Use an ECDSA key

ISSUE CERTIFICATES (cont.)

`mkcert -pkcs12 <name>`
Also produce a .p12 bundle
Default password is changeit.

`mkcert -p12-file <f> -pkcs12 <name>`
Set .p12 output path

`mkcert -client <name>`
Client-authentication cert

`mkcert <user>@<domain>`
S/MIME cert for an email address

`mkcert -csr <file>`
Sign an existing CSR
Cannot be combined with names or other flags except -install and -cert-file.

TRUST ELSEWHERE

`CAROOT=<dir> mkcert -install`
Install a copied CA on another host
Copy only rootCA.pem there; the key is not needed to trust.

`CAROOT=<dir> mkcert <name>`
Use a separate CA

`NODE_EXTRA_CA_CERTS="$(mkcert -CAROOT)/rootCA.pem"`
Make Node.js trust the CA

TRUST ELSEWHERE (cont.)

`$(mkcert -CAROOT)/rootCA.pem`
Root cert to import on phones
iOS: install the profile, then enable full trust in settings.

`JAVA_HOME=<jdk> mkcert -install`
Trust CA in Java keystore

RECIPES

Local HTTPS for a dev server

```
mkcert -install
mkcert -cert-file dev.pem -key-file dev-key.pem localhost 127.0.0.1 ::1
# point the server at dev.pem and dev-key.pem
```

nginx server block

```
mkcert -cert-file dev.pem -key-file dev-key.pem app.test
# ssl_certificate     /path/dev.pem;
# ssl_certificate_key /path/dev-key.pem;
curl https://app.test
```

Share the CA with a teammate or VM

```
cp "$(mkcert -CAROOT)/rootCA.pem" ./
# on the other machine, with rootCA.pem in <dir>:
CAROOT=<dir> mkcert -install
```

Inspect the result

```
openssl x509 -in dev.pem -noout -subject -issuer -dates -ext subjectAltName
```