Search

Docs Home
Quickstart
CockroachDB Cloud
CockroachDB Self-Hosted
Reference
SQL Playground

Version v21.1.11

Create Security Certificates using Openssl

Contribute

To secure your CockroachDB cluster's inter-node and client-node communication, you need to provide a Certificate Authority (CA) certificate that has been used to sign keys and certificates (SSLs) for:

- Nodes
- Clients
- DB Console (optional)

To create these certificates and keys, use the `cockroach cert` commands with the appropriate subcommands and flags, use `openssl` commands, or use a custom CA (for example, a public CA or your organizational CA).

Use cockroach cert | **Use Openssl** | Use custom CA

Subcommands

Subcommand	Usage
`openssl genrsa`	Create an RSA private key.
`openssl req`	Create CA certificate and CSRs (certificate signing requests).
`openssl ca`	Create node and client certificates using the CSRs.

Configuration files

To use `openssl req` and `openssl ca` subcommands, you need the following configuration files:

File name pattern	File usage
`ca.cnf`	CA configuration file
`node.cnf`	Server configuration file
`client.cnf`	Client configuration file

Certificate directory

To create node and client certificates using the OpenSSL commands, you need access to a local copy of the CA certificate and key. We recommend creating all certificates (node, client, and CA certificates), and node and client keys in one place and then distributing them appropriately. Store the CA key somewhere safe and keep a backup; if you lose it, you will not be able to add new nodes or clients to your cluster.

Required keys and certificates

Use the `openssl genrsa` and `openssl req` subcommands to create all certificates, and node and client keys in a single directory, with the files named as follows:

Node key and certificates

File name pattern	File usage

On this page

File name pattern	File usage
`ca.crt`	CA certificate
`node.crt`	Server certificate
`node.key`	Key for server certificate

Client key and certificates

File name pattern	File usage
`ca.crt`	CA certificate.
`client.<user>.crt`	Client certificate for `<user>` (for example: `client.root.crt` for user `root`).
`client.<user>.key`	Key for the client certificate.

Note the following:

- The CA key should not be uploaded to the nodes and clients, so it should be created in a separate directory.
- Keys (files ending in `.key`) must not have group or world permissions (maximum permissions are 0700, or `rwx------`). This check can be disabled by setting the environment variable `COCKROACH_SKIP_KEY_PERMISSION_CHECK=true`.

Examples

Step 1. Create the CA key and certificate pair

1. Create two directories:

```
$ mkdir certs my-safe-directory
```

   - `certs`: Create your CA certificate and all node and client certificates and keys in this directory and then upload the relevant files to the nodes and clients.
   - `my-safe-directory`: Create your CA key in this directory and then reference the key when generating node and client certificates. After that, keep the key safe and secret; do not upload it to your nodes or clients.

2. Create the `ca.cnf` file and copy the following configuration into it.

   You can set the CA certificate expiration period using the `default_days` parameter. We recommend using the CockroachDB default value of the CA certificate expiration period, which is 365 days.

```
# OpenSSL CA configuration file
[ ca ]
default_ca = CA_default

[ CA_default ]
default_days = 365
database = index.txt
serial = serial.txt
default_md = sha256
copy_extensions = copy
unique_subject = no

# Used to create the CA certificate.
[ req ]
prompt=no
distinguished_name = distinguished_name
```

```
x509_extensions = extensions

[ distinguished_name ]
organizationName = Cockroach
commonName = Cockroach CA

[ extensions ]
keyUsage = critical,digitalSignature,nonRepudiation,keyEncipherment,keyCertSign
basicConstraints = critical,CA:true,pathlen:1

# Common policy for nodes and users.
[ signing_policy ]
organizationName = supplied
commonName = optional

# Used to sign node certificates.
[ signing_node_req ]
keyUsage = critical,digitalSignature,keyEncipherment
extendedKeyUsage = serverAuth,clientAuth

# Used to sign client certificates.
[ signing_client_req ]
keyUsage = critical,digitalSignature,keyEncipherment
extendedKeyUsage = clientAuth
```

Warning:
The `keyUsage` and `extendedkeyUsage` parameters are vital for CockroachDB functions. You can modify or omit other parameters as per your preferred OpenSSL configuration and you can add additional usages, but do not omit `keyUsage` and `extendedkeyUsage` parameters or remove the listed usages.

3. Create the CA key using the `openssl genrsa` command:

```
$ openssl genrsa -out my-safe-directory/ca.key 2048
```

```
$ chmod 400 my-safe-directory/ca.key
```

4. Create the CA certificate using the `openssl req` command:

```
$ openssl req \
-new \
-x509 \
-config ca.cnf \
-key my-safe-directory/ca.key \
-out certs/ca.crt \
-days 365 \
-batch
```

5. Reset database and index files:

```
$ rm -f index.txt serial.txt
```

```
$ touch index.txt
```

```
$ echo '01' > serial.txt
```

Step 2. Create the certificate and key pairs for nodes

In the following steps, replace the placeholder text in the code with the actual username and node address.

1. Create the `node.cnf` file for the first node and copy the following configuration into it:

```
# OpenSSL node configuration file
[ req ]
prompt=no
distinguished_name = distinguished_name
req_extensions = extensions

[ distinguished_name ]
organizationName = Cockroach

[ extensions ]
subjectAltName = critical,DNS:<node-hostname>,DNS:<node-domain>,IP:<IP Address>
```

> **Warning:**
> The `subjectAltName` parameter is vital for CockroachDB functions. You can modify or omit other parameters as per your preferred OpenSSL configuration, but do not omit the `subjectAltName` parameter.

2. Create the key for the first node using the `openssl genrsa` command:

```
$ openssl genrsa -out certs/node.key 2048
```

```
$ chmod 400 certs/node.key
```

3. Create the CSR for the first node using the `openssl req` command:

```
$ openssl req \
-new \
-config node.cnf \
-key certs/node.key \
-out node.csr \
-batch
```

4. Sign the node CSR to create the node certificate for the first node using the `openssl ca` command.

```
$ openssl ca \
-config ca.cnf \
```

```
-keyfile my-safe-directory/ca.key \
-cert certs/ca.crt \
-policy signing_policy \
-extensions signing_node_req \
-out certs/node.crt \
-outdir certs/ \
-in node.csr \
-batch
```

5. Verify the values in the `Subject Alternative Name` field in the certificate:

```
$ openssl x509 -in certs/node.crt -text | grep "X509v3 Subject Alternative Name" -A 1
```

Sample output:

```
X509v3 Subject Alternative Name: critical
            DNS:localhost, DNS:node.example.io, IP Address:127.0.0.1
```

Step 3. Create the certificate and key pair for the first user

In the following steps, replace the placeholder text in the code with the actual username.

1. Create the `client.cnf` file for the first user and copy the following configuration into it:

```
[ req ]
prompt=no
distinguished_name = distinguished_name
req_extensions = extensions

[ distinguished_name ]
organizationName = Cockroach
commonName = <username_1>

[ extensions ]
subjectAltName = DNS:root
```

> **Warning:**
> The `commonName` and `subjectAltName` parameters are vital for CockroachDB functions. You can modify or omit other parameters as per your preferred OpenSSL configuration, but do not omit the `commonName` parameter or modify the `subjectAltName` parameter.

2. Create the key for the first client using the `openssl genrsa` command:

```
$ openssl genrsa -out certs/client.<username_1>.key 2048
```

```
$ chmod 400 certs/client.<username_1>.key
```

3. Create the CSR for the first client using the `openssl req` command:

```
$ openssl req \
-new \
-config client.cnf \
-key certs/client.<username_1>.key \
-out client.<username_1>.csr \
-batch
```

4. Sign the client CSR to create the client certificate for the first client using the `openssl ca` command.

```
$ openssl ca \
-config ca.cnf \
-keyfile my-safe-directory/ca.key \
-cert certs/ca.crt \
-policy signing_policy \
-extensions signing_client_req \
-out certs/client.<username_1>.crt \
-outdir certs/ \
-in client.<username_1>.csr \
-batch
```

5. Verify the values in the `CN` field in the certificate:

```
$ openssl x509 -in certs/client.<username_1>.crt -text | grep CN=
```

Sample Output:

```
Issuer: O=Cockroach, CN=Cockroach CA
    Subject: O=Cockroach, CN=maxroach
```

Step 4. Start a local cluster and connect using a connection URL

1. Start a single-node cluster:

```
$ cockroach start-single-node --certs-dir=certs --cert-principal-map=<node-domain>:node,<username_1>:root --background
```

2. Connect to the cluster using a connection URL:

```
$ cockroach sql --url='postgres://<hostname>:26257/?sslmode=verify-full&sslrootcert=certs/ca.crt&sslcert=certs/client.<username_1>.crt&ssl
```

3. Create a new SQL user:

```
> create user <username_2>;
```

```
> \q
```

Step 5. Create the certificate and key pair for a client

In the following steps, replace the placeholder text in the code with the actual username.

1. Edit the `client.cnf` file for the client and copy the following configuration into it:

```
[ req ]
prompt=no
distinguished_name = distinguished_name

[ distinguished_name ]
organizationName = Cockroach
commonName = <username_2>
```

> **Warning:**
> The `commonName` parameter is vital for CockroachDB functions. You can modify or omit other parameters as per your preferred OpenSSL configuration, but do not omit the `commonName` parameter.

2. Create the key for the first client using the `openssl genrsa` command:

```
$ openssl genrsa -out certs/client.<username_2>.key 2048
```

```
$ chmod 400 certs/client.<username_2>.key
```

3. Create the CSR for the first client using the `openssl req` command:

```
$ openssl req \
-new \
-config client.cnf \
-key certs/client.<username_2>.key \
-out client.<username_2>.csr \
-batch
```

4. Sign the client CSR to create the client certificate for the first client using the `openssl ca` command.

```
$ openssl ca \
-config ca.cnf \
-keyfile my-safe-directory/ca.key \
-cert certs/ca.crt \
-policy signing_policy \
-extensions signing_client_req \
-out certs/client.<username_2>.crt \
-outdir certs/ \
-in client.<username_2>.csr \
-batch
```

5. Verify the values in the `CN` field in the certificate:

```
$ openssl x509 -in certs/client.<username_2>.crt -text | grep CN=
```

Sample output:

```
Issuer: O=Cockroach, CN=Cockroach CA
    Subject: O=Cockroach, CN=roach
```

6. Connect to the SQL client using the client certificate:

```
$ cockroach sql --url='postgres://<username_2>@<hostname>:26257/?sslmode=verify-full&sslrootcert=certs/ca.crt&sslcert=certs/client.<usern
```

For each node in your deployment, repeat Step 2 and upload the CA certificate and node key and certificate to the node. For each client, repeat Step 5 and upload the CA certificate and client key and certificate to the client.

After you have uploaded all the keys and certificates to the corresponding nodes and clients, remove the `.pem` files in the `certs` directory. These files are unnecessary duplicates of the `.crt` files that CockroachDB requires.

See also

- Manual Deployment: Learn about starting a multi-node secure cluster and accessing it from a client.
- Start a Node: Learn more about the flags you pass when adding a node to a secure cluster
- Client Connection Parameters