

Задание 1

Необходимо создать CA и выпустить клиентские и серверные сертификаты.

- Выпустить CA
- Выпустить серверный сертификат (подписанный CA) с альтернативным именем: foo.bar.baz
- Выпустить клиентский сертификат

Road to 10: Развернуть nginx server с использованием сертификатов

Выпуск сертификатов находится в script.sh

Проверка подписи находится в verify_cert.sh

Проверка альтернативного имени в verify_host.sh

1. Выпуск самоподписанного CA

```
CERTS_DIR="./certs"
openssl genrsa -out "$CERTS_DIR/ca.key" 4096
openssl req -x509 -new -nodes -key "$CERTS_DIR/ca.key" -sha256 -days 3650 \
    -out "$CERTS_DIR/ca.crt" \
    -subj "/CN=MyRootCA"
```

2. Выпуск серверного сертификата, подписанного CA с альтернативным именем foo.bar.baz. Конфигурация в файле server.cnf

```
cat > "$CERTS_DIR/server.cnf" <<'EOF'
[req]
distinguished_name = dn
req_extensions = req_ext
prompt = no

[dn]
CN = foo.bar.baz
```

```

[req_ext]
subjectAltName = @alt_names
basicConstraints = CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[alt_names]
DNS.1 = foo.bar.baz
EOF

openssl genrsa -out "$CERTS_DIR/server.key" 2048
openssl req -new -key "$CERTS_DIR/server.key" -out "$CERTS_DIR/se
rver.csr" -config "$CERTS_DIR/server.cnf"
openssl x509 -req -in "$CERTS_DIR/server.csr" -CA "$CERTS_DIR/ca.c
rt" -CAkey "$CERTS_DIR/ca.key" -CAcreateserial \
-out "$CERTS_DIR/server.crt" -days 365 -sha256 -extfile "$CERTS_DI
R/server.cnf" -extensions req_ext

```

3. Выпуск клиентского сертификата, подписанного CA. Конфигурация в файле client.cnf

```

cat > "$CERTS_DIR/client.cnf" <<'EOF'
basicConstraints = CA:FALSE
keyUsage = digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth
EOF

openssl genrsa -out "$CERTS_DIR/client.key" 2048
openssl req -new -key "$CERTS_DIR/client.key" -out "$CERTS_DIR/clie
nt.csr" \
-subj "/CN=TestClient"
openssl x509 -req -in "$CERTS_DIR/client.csr" -CA "$CERTS_DIR/ca.cr
t" -CAkey "$CERTS_DIR/ca.key" -CAcreateserial \
-out "$CERTS_DIR/client.crt" -days 365 -sha256 -extfile "$CERTS_DI
R/client.cnf"

```

4. Проверка сертификатов (вынесла в скрипт verify_crt.sh)

```
openssl verify -CAfile "$CERTS_DIR/ca.crt" "$CERTS_DIR/server.crt"
openssl verify -CAfile "$CERTS_DIR/ca.crt" "$CERTS_DIR/client.crt"
```

5. Проверка альтернативного имени (вынесла в скрипт verify_host.sh)

```
openssl verify -CAfile "$CERTS_DIR/ca.crt" -verify_hostname foo.bar.baz "$CERTS_DIR/server.crt"
```

Конфигурация для nginx (./config/nginx.conf)

```
events {}

http {
    server {
        listen 443 ssl;
        server_name foo.bar.baz;

        ssl_certificate      /etc/nginx/certs/server.crt;
        ssl_certificate_key   /etc/nginx/certs/server.key;

        ssl_client_certificate /etc/nginx/certs/ca.crt;
        ssl_verify_client     on;

        location / {
            root /usr/share/nginx/html;
            index index.html;
        }
    }
}
```

compose файл для запуска контейнера nginx:

```
services:
  nginx:
    image: nginx:alpine
    container_name: nginx-sem1
```

ports:

- "443:443"

volumes:

- ./config/nginx.conf:/etc/nginx/nginx.conf:ro
- ./certs:/etc/nginx/certs:ro
- ./html:/usr/share/nginx/html:ro

restart: unless-stopped

1. Запускаем скрипт script.sh, он выпускает са, серверный и клиентский сертификаты. в конце есть проверка на то, что клиентский и серверный сертификаты подписаны са. в серверный сертификат прокидываю конфигурацию с альтернативным именем, операциями, которые можно делать с ключом (KeyUsage) и extendedKeyUsage = serverAuth, показывающий, что сертификат используется для серверной аутентификации. то же самое делаю с клиентским сертификатом: прокидываю конфигурацию, указываю, что сертификат будет использоваться для клиентской аутентификации.

```
(base) vikas@MacBook-Air-Viktoria sem1 % chmod +x script.sh
(base) vikas@MacBook-Air-Viktoria sem1 % ./script.sh
Certificate request self-signature ok
subject=CN = foo.bar.baz
Certificate request self-signature ok
subject=CN = TestClient
./certs/server.crt: OK
./certs/client.crt: OK
(base) vikas@MacBook-Air-Viktoria sem1 %
```



ca.crt



ca.key



ca.srl



client.cnf



client.crt



client.csr



client.key



server.cnf



server.crt



server.csr



server.key

- Запускаем скрипт проверки подписи (он уже есть в основном скрипте, ну чисто на всякий случай):

```
(base) vikas@MacBook-Air-Viktoria sem1 % chmod +x verify_srt.sh
(base) vikas@MacBook-Air-Viktoria sem1 % ./verify_srt.sh
./certs/server.crt: OK
./certs/client.crt: OK
(base) vikas@MacBook-Air-Viktoria sem1 %
```




- Запускаем скрипт проверки альтернативного имени

```
(base) vikas@MacBook-Air-Viktoria sem1 % chmod +x verify_host.sh
(base) vikas@MacBook-Air-Viktoria sem1 % ./verify_host.sh
./certs/server.crt: OK
(base) vikas@MacBook-Air-Viktoria sem1 %
```

- Поднимаем докер контейнер с nginx

```
docker-compose up -d
```

```
(base) vikas@MacBook-Air-Viktoria sem1 % docker-compose up -d
[+] Running 1/2
  ⚡ Network sem1_default Created                                0.2s
  ✔ Container nginx-sem1 Started                                0.2s
(base) vikas@MacBook-Air-Viktoria sem1 %
```

<input type="checkbox"/>	<div><div></div><div><div>sem1</div></div></div>	Running (1/1)	0%	22 seconds ago	<div><div></div><div>:</div><div></div></div>
<input type="checkbox"/>	<div><div></div><div><div>nginx-sem1</div><div>03fac64c633f </div></div></div> <div>nginx:alpine</div>	Running	0% 443:443	22 seconds ago	<div><div></div><div>:</div><div></div></div>

5. Делаем курл запрос без сертификата

```
curl -vk https://foo.bar.baz/
```

Ответ:

```
* Host foo.bar.baz:443 was resolved.
* IPv6: (none)
* IPv4: 127.0.0.1
* Trying 127.0.0.1:443...
* Connected to foo.bar.baz (127.0.0.1) port 443
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Request CERT (13):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Certificate (11):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384 / X25519 / RSASSA-PSS
* ALPN: server accepted http/1.1
* Server certificate:
* subject: CN=foo.bar.baz
* start date: Sep 12 13:18:15 2025 GMT
* expire date: Sep 12 13:18:15 2026 GMT
* issuer: CN=MyRootCA
* SSL certificate verify result: unable to get local issuer certificate (20),
```

continuing anyway.

* Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption

* using HTTP/1.x

> GET / HTTP/1.1

> Host: foo.bar.baz

> User-Agent: curl/8.9.1

> Accept: */*

>

* Request completely sent off

* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):

* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):

< HTTP/1.1 400 Bad Request

< Server: nginx/1.29.1

< Date: Fri, 12 Sep 2025 13:29:30 GMT

< Content-Type: text/html

< Content-Length: 237

< Connection: close

<

<html>

<head><title>400 No required SSL certificate was sent</title></head>

<body>

<center><h1>400 Bad Request</h1></center>

<center>No required SSL certificate was sent</center>

<hr><center>nginx/1.29.1</center>

</body>

</html>

* shutting down connection #0

* TLSv1.3 (OUT), TLS alert, close notify (256):

6. Делаем курл запрос с сертификатом

```
curl -vk --cert certs/client.crt --key certs/client.key https://foo.bar.baz/
```

Ответ:

* Host foo.bar.baz:443 was resolved.

* IPv6: (none)

```

* IPv4: 127.0.0.1
* Trying 127.0.0.1:443...
* Connected to foo.bar.baz (127.0.0.1) port 443
* ALPN: curl offers h2,http/1.1
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Request CERT (13):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Certificate (11):
* TLSv1.3 (OUT), TLS handshake, CERT verify (15):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384 / X25519 / RSASSA-PSS
* ALPN: server accepted http/1.1
* Server certificate:
* subject: CN=foo.bar.baz
* start date: Sep 12 13:18:15 2025 GMT
* expire date: Sep 12 13:18:15 2026 GMT
* issuer: CN=MyRootCA
* SSL certificate verify result: unable to get local issuer certificate (20), continuing anyway.
* Certificate level 0: Public key type RSA (2048/112 Bits/secBits), signed using sha256WithRSAEncryption
* using HTTP/1.x
> GET / HTTP/1.1
> Host: foo.bar.baz
> User-Agent: curl/8.9.1
> Accept: */*
>
* Request completely sent off
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
< HTTP/1.1 200 OK
< Server: nginx/1.29.1

```



```
< Date: Fri, 12 Sep 2025 13:31:49 GMT
< Content-Type: text/html
< Content-Length: 11
< Last-Modified: Wed, 10 Sep 2025 17:27:06 GMT
< Connection: keep-alive
< ETag: "68c1b4ea-b"
< Accept-Ranges: bytes
<
* Connection #0 to host foo.bar.baz left intact
<h1>OK</h1>%
```