Client Machine

- Init cert script
    - Automation of script to read serial number from assigned serial number in text file,
    - generate client certificate and key,
    - generate CSR using key and configuration file,
    - connect to secure shell server on local host and designated port using SSH user and key,
    - check if remote directory exists, then using SCP, copy CSR to remote directory,
    - sleep for 20 seconds to wait on certificate signing process,
    - check for signed certificate if it has been generated for an amount of 10 times,
    - if check is true, copy it using SCP from remote server to local filesystem,
    - access NGINX HTTPS endpoint to verify client certificate and achieve mTLS by using client key and signed certificate, if not, client certificate is not trusted,
    - Init script terminates after successful mTLS,

Production PC

- Generate verification service certs
    - Generates server private key,
    - generates server CSR using key,
    - creates server certificate configuration file,
    - generates server certificate after being signed by RootCA to be used for mTLS,
    - signed certificate would then be saved in server volume for HTTPS endpoint use,
- Docker Containers
    - Transfer Service
        - Configures SSH daemon,
        - Creates directory for mounted volume,
        - Set permissions for SSH directory,
        - Copy SSH daemon configuration,
        - Expose SSH port,
        - Start the server,
    - Signing Service
        - Exports environment variables,
        - Watches remote directory for new CSRs,

        - Activates once new CSR detected,
        - Awaits Commercial Serial Number text file to contain serial number,
        - After checking for it, it extracts and saves to variable,
        - CSR is then signed with RootCA key and certificate,
        - Original and Commerical serial number are saved in a CSV file,
        - Client certificate is outputted into remote directory,
    - Verification Service
        - Initiates NGINX HTTPS endpoint
- Docker Compose
    - Runs all three services,
    - Assigns ports and volumes

- *Instructions for Setup*
    - Username and password for any sudo commands: production
    - Make sure docker engine is running by running commands:
        - sudo systemctl status docker
        - if not running, sudo systemctl start docker
    - Generate rootCA on Production PC under path DEV/server/server-volumes/root-ca **skip if rootCA already setup*
        - *openssl ecparam -genkey -name prime256v1 -noout -out rootCA.key (generates private key)*
        - *openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 3650 -out rootCA.pem *
          -subj "/C=DE/O=YUKAWA/CN=nlcROOTCA" (generates self-signed certificate)
    - Run generate-verification-service-certs.sh on Production PC to create server-side certificate for mTLS in terminal under path of DEV folder **skip if already generated*
    - *Run # chmod 600 -R ./server-volumes/authorized_keys under path DEV/server*
    - Under the right directory path (DEV/server) where Container files will be located, run these commands to activate them:
        - docker-compose -f server.docker-compose.yml build
        - docker-compose -f server.docker-compose.yml up -d
        - docker-compose -f server.docker-compose.yml ps (verifies if containers are running)
        - docker logs -f containernumber (to display logs of cert signing container)
    - Run QR code keyboard emulator to input commercial serial number in a separate dedicated text file which is cleared upon every certificate signing.

        - In a separate terminal under right directory path (text file is in cs server volume directory server/server-volumes/cs) keyboard should type this command: echo Commercial Serial Number > commercial_serial.txt
        - It is also possible to do this manually by creating a text file with same name, and manually adding serial number
    - Assign CM4 serial number in client/serial_number.txt
    - In separate terminal under path DEV/client, run ./init-cert-script.sh
    - To stop server, run: docker-compose -f server.docker-compose.yml down

- *Network Requirements*
    - Private WLAN created by Wi-Fi router (no internet required)
    - Client and Server machines connected wirelessly or with Ethernet
    - Reserve static IP Address for Production PC via router's DHCP reservation settings, so client initiates SSH connection with production pc using its static host IP address, and for HTTPS endpoint, too -> instead of using localhost/local IP use the static ip of PC

```
-   SSH_HOST="127.0.0.1"
-   curl -k https://localhost
both are located in init-cert-script.sh
```

    - To test connection, run ping Production PC IP on client machine, should return PONG if successful