# Beispieldokumentation

## Deutsche Beschreibung

NUTZUNGSBEDINGUNGEN

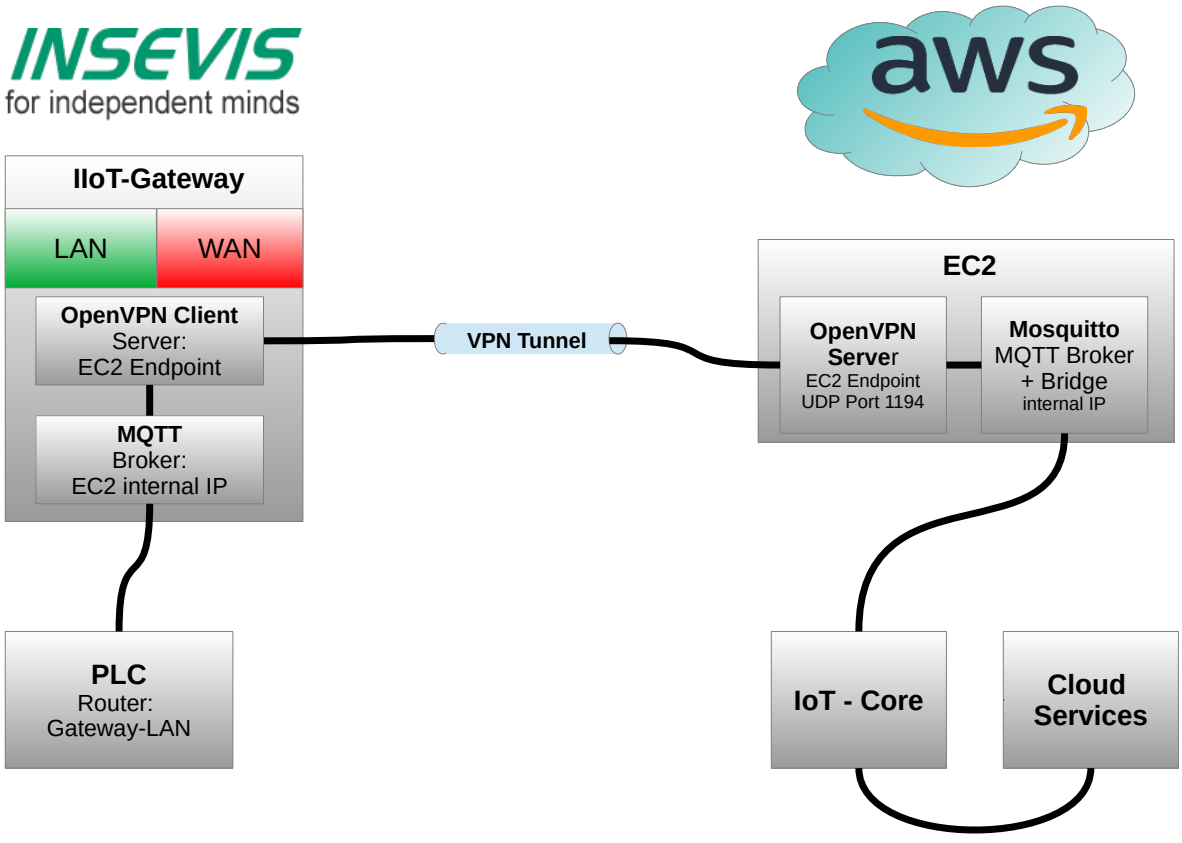
Die Verwendung der Beispielprogramme erfolgt ausschließlich unter Anerkennung folgender Bedingungen durch den Benutzer:

INSEVIS bietet kostenlose Beispielprogramme für die optimale Nutzung der S7-Programmierung und zur Zeitersparnis bei der Programmerstellung. Für direkte, indirekte oder Folgeschäden des Gebrauchs dieser Software schließt INSEVIS jegliche Gewährleistung genauso aus, wie die Haftung für alle Schäden, die aus die aus der Weitergabe der die Beispielinformationen beinhaltenden Software resultieren.

**BEISPIELBESCHREIBUNG AWS**

**Ziel**

Die AWS IoT-Core-Dienste stellen Anforderungen an die Übertragungssicherheit, die im INSEVIS-IIoT-Gateway im MQTT-Dienst noch nicht implementiert sind.
Zur sicheren Übertragung dient ein VPN-Tunnel in einen Virtuellen Server (AWS CE2) in dem eine MQTT-Bridge die Daten an IoT-Core den Anfordrungen entsprechend weiterleitet.
Dieser VPN-Tunnel kann gleichzeitig zur Fernwartung genutzt werden.




**Voraussetungen und Vorbetrachtung**

- Kenntnisse im Umgang mit Linux-Console und INSEVIS IIoT-Gateway
- ein AWS Account und Kenntnisse zur Konfiguration dessen,
- ein PC mit ssh und scp
- Planung der Applikation (Nutzdaten: Struktur, Namen, Richtung)

**AWS IoT-Core**

- ein "Thing" anlegen und ein SDK downloaden

  → im SDK befindet sich das Schlüsselpaar (es geht bestimmt auch anders)

- das Root-Zertifikat hier oder in der EC2 downloaden:

```
$ wget https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

  (es gibt auch andere Pfade)

- die Policy einrichten
  (per default sind ggf. nur die demo-topics und der demo-client-name des SDKs freigegeben !)
  Einrichten der topic- und connect-Filter

- IoT-Core Endpoint "merken"

**AWS EC2 konfigurieren**

z.B. „micro“, „debian“ „PC“

- RSA-Schlüsselpaar für SSH erstellen und download
- externe IP-Adresse (EC2-Endpoint-IP) "merken"
- am PC ssh mit AWS-Key einrichten (Datei .ssh mit Pfad auf den private key)

  → vom PC auf EC2 einloggen mit

```
> ssh admin@<EC2-Endpoint-IP>
```

- Die interne IP-Adresse ist im login-promt, oder ausgeben mit

```
$ ip add
```

  → interne IP-Adresse „merken“

- MQTT-Broker installieren:

```
$ sudo apt install mosquitto
$ sudo apt install mosquitto-clients (nur zu Testzwecken)
```

- MQTT-Broker konfigurieren:

```
$ sudo nano /etc/mosquitto/mosquitto.conf
```

  per default ist nur „localhost“ erlaubt, hinzufügen:

```
listener 1883
```

- die 2 Schlüssel aus dem IoT-Core-SDK vomm PC in die EC2 verschieben:

```
> scp keyfile admin@<EC2-Endpoint-IP>:~
```

- das Rootzertifiakt holen (oder - s. oben - mit den anderen Keys per scp)

```
$ sudo wget https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

- die 3 Schlüssel nach /etc/mosquitto/cert schieben (in 2 Schritten, weil über scp kein sudo-Zugriff):

```
$ sudo mv <keyfile> /etc/mosquitto/certs
```

- MQTT-Bridge konfigurieren:
  Datei „bridge.conf“ nach /etc/mosquitto/conf.d/ kopieren oder dort anlegen:

```
$ sudo nano /etc/mosquitto/conf.d/bridge.conf
```

  anpassen:

```
address       <IoT-Core Endpoint>
topic         <ergibt sich aus der Applikation> in 1
topic         <ergibt sich aus der Applikation> out 1
```

  restart:

```
$sudo systemctl restart mosquitto
```

Jetzt könnte man den Mosquitto schon lokal testen und die Bridge über die TestSeite im IoT-Core

```
$ mosquitto_sub demotopic/#
$ mosquitto_pub demotopic/text blabla
```

- openVPN installieren:
  **`$ sudo apt install openvpn`**
  (Konfiguration kann erst später erfolgen)

**INSEVIS IIoT Gateway**

Für die geplante Anwendung ist die SPS im IIoT-Gateway als
- Connection anzulegen und dazu die entsprechenden
- Datenpunkte.
  (Danach sollte der Status der Connection „running“ sein)

- openVPN CA-Key erstellen
- als "PublicIP" die Externe IP der EC2 (EC2-Endpoint-IP) eintragen (+ „Save to Device“)
- openVPN-Server generieren:
  Option Server-Netz routen enablen
- einen oder mehrere openVPN-clients anlegen und exportieren
  (e.g. IIoTGateway1..3, SupportPC)
  Client-Netz routen wenn gewünscht

*Hinweis: openVPN-Server NICHT starten, der wird hier NICHT benutzt*

- ein „Complete Backup“ erstellen,
  <file>.bup in <file>.tgz umbenennen,
  tar-Archiv öffnen und Unterverzeichnis "./openvpn/" entpacken (Der Rest kann weg.)

- Datei ./openvpn/server.conf korrigieren:
  push "route 172.31.44.155" sollte die lokale IP der EC2 enthalten
  (da wird aber die IP-Adresse des Gateways drin stehen)
  route 192.168.75.0 255.255.255.0
  diese Zeile(n) ermöglichen Zugriff aus der EC2 „hinter“ das Gateway (z.B. auf die SPS)

*Hinweis: Diese Konfiguration eines openVPN kann man auch an jedem Linux-PC oder Raspberry Pi mit "Bordmitteln" machen. Wir benutzen hier die Web-Oberfläche des Gateways.*

- Die Konfiguration im Verzeichnis ./openVPN/ sollte nun aus /ccd, /easy-rsa und server.conf bestehen
  Diese Konfiguration vom PC in die EC2 laden:
  **`> scp -r openvpn/ admin@`***`<EC2-Endpoint-IP>`***`:~`**

- openVPN-Client aktivieren:
  Den für dieses IIoTGateway erstellen openVPN-client über "Import openVPN Client"
  vom eben exportieren File "zurück-importieren".
  [Start], "Start at startup" und "save to device" nicht vergessen.

  → das Gateway zeigt jetzt "running", aber der Server fehlt noch - das sieht man aber nur im Logfile

- IIoT-Gateway-MQTT-Konfiguration:
  Broker:
  Für die getunnelte MQTT-Kommunikation ist als Broker die interne EC2-IP-Adresse zu benutzen.
(Das funktioniert aber jetzt noch nicht !!)
  Datenpunkte:
  Datenpunkte und zugeordnete topics sind entspechend der definierten Anwendung einzutragen.
  „String conversion“ muss aktiviert werden.

Zum Test kann der EC2-Endpoint (mit Freigabe UDP port 1883) oder jeder andere „offene“ Test-MQTT-Broker dienen. (Damit sollte der Status der Connection „running“ sein)

**AWS EC2 Teil 2**

- in den EC2-Sicherheitseinstellungen für eingehenden Datenverkehr UDP Port 1194 freigeben

- die eben vom PC übertragene openVPN-Konfiguration ins System verschieben:

```
$ sudo mv ~/openvpn/ccd/         /etc/openvpn
$ sudo mv ~/openvpn/easy-rsa/    /etc/openvpn
$ sudo mv ~/openvpn/server.conf /etc/openvpn
```

- openVPN-Dienst starten:

```
$ sudo systemctl enable openvpn@server.service
$ sudo systemctl start  openvpn@server.service
```

- openVPN testen mit

```
$ ping 10.1.0.1                          = selbst-Ping des servers
$ ping 10.1.0.2   (oder .3)              = Client-Ping des servers
$ ping <IIoT-Gateway-LAN-Adresse>        = Ping des gerouteten Clients
```

**Fernwartungsoption**

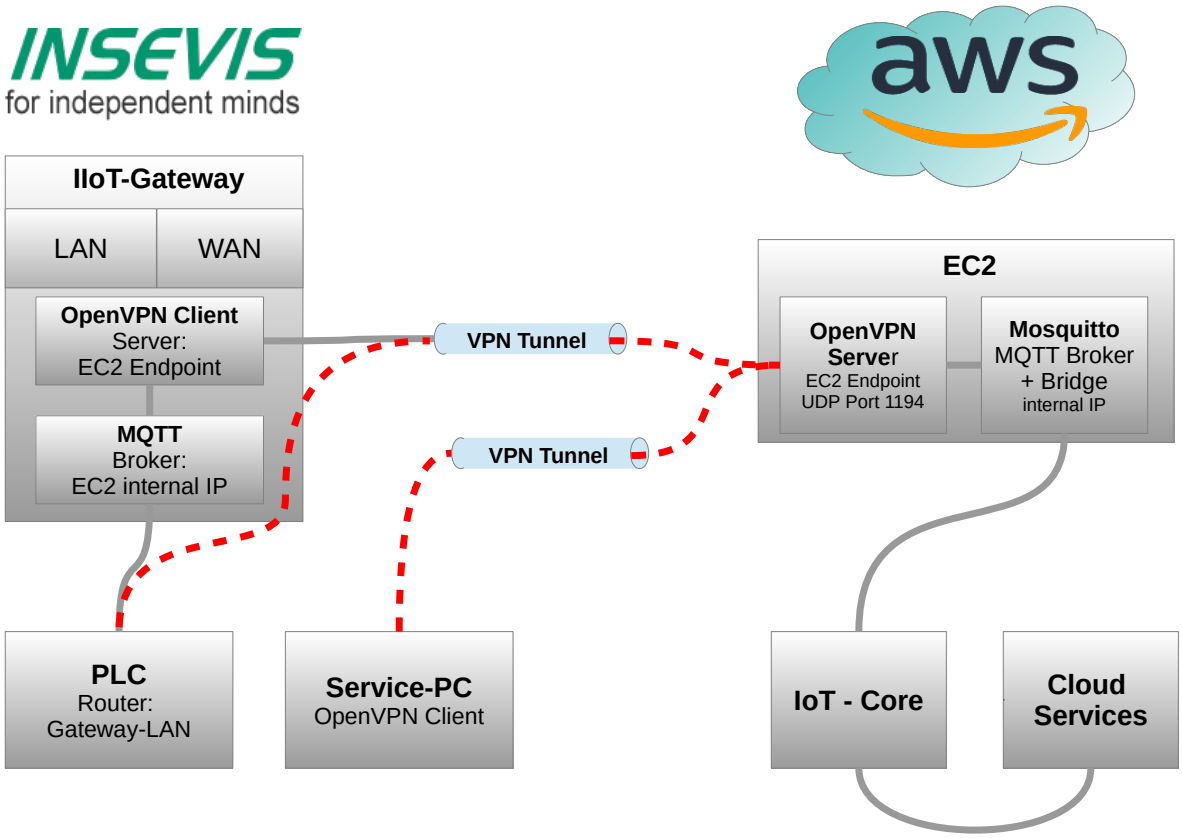
Wurde ein weiterer VPN-Client generiert, kann auf einem PC mit installiertem openVPN-client das ovpn-File aus dem exportierten Paket impotiert werden (ohne Nutzername/Passwort).
Die Zielnetzadresse (IIoT-Gateway-LAN) muss dem PC bekannt gegeben werden. Man kann das routing im PC manuell setzen oder in das ovpn-File einfügen:

```
route 192.168.80.0 255.255.255.0
```

Für ein site-to-site-VPN muss in der EC2 das ip-forwarding eingeschaltet werden:

```
$ echo 1 | sudo tee -a /proc/sys/net/ipv4/ip_forward
```

Jetzt sind die Konfigurationsoberfläche des IIoT-Gateways und die dahinterliegende SPS über VPN erreichbar.
Wichtig ist, dass sich keine IP-Adressbereiche der lokalen Netze (IIoT-Gateway-LAN/-WAN, PC, EC2) überschneiden.
Da EC2 als openVPN-server öffentlich erreichbar ist, muss keine lokale Firewall konfiguriert werden.



**Inbetriebnahme**
Wenn sich S7-Variablen ändern, sollte eine MQTT-Message in den EC2-Broker gesendet werden.
Das kann – ohne bridge-Funktion – in der EC2-commandline bebachtet werden:

```
$ mosquitto_sub -t <mein topic>
```

Ist die Bridge richtig eingerichtet, erscheinen die Nachrichten auch im AWS-IoT-Core MQTT-Testclient.
Dort kann man auch Nachrichten senden, die dann in der EC2 sowie in der SPS ankommen sollten.

**Hinweis:**
- noch keine -

RÜCKMELDUNGEN
Möchten Sie Erweiterungswünsche oder Fehler zu diesen Beispielen melden oder wollen Sie anderen eigene Bespielprogramme kostenlos zur Verfügung stellen? ***Bitte informieren Sie uns unter info@insevis.de***
Gern werden Ihre Programme -auf Wunsch mit Benennung des Autors- allen INSEVIS- Kunden zur Verfügung gestellt.

# English description

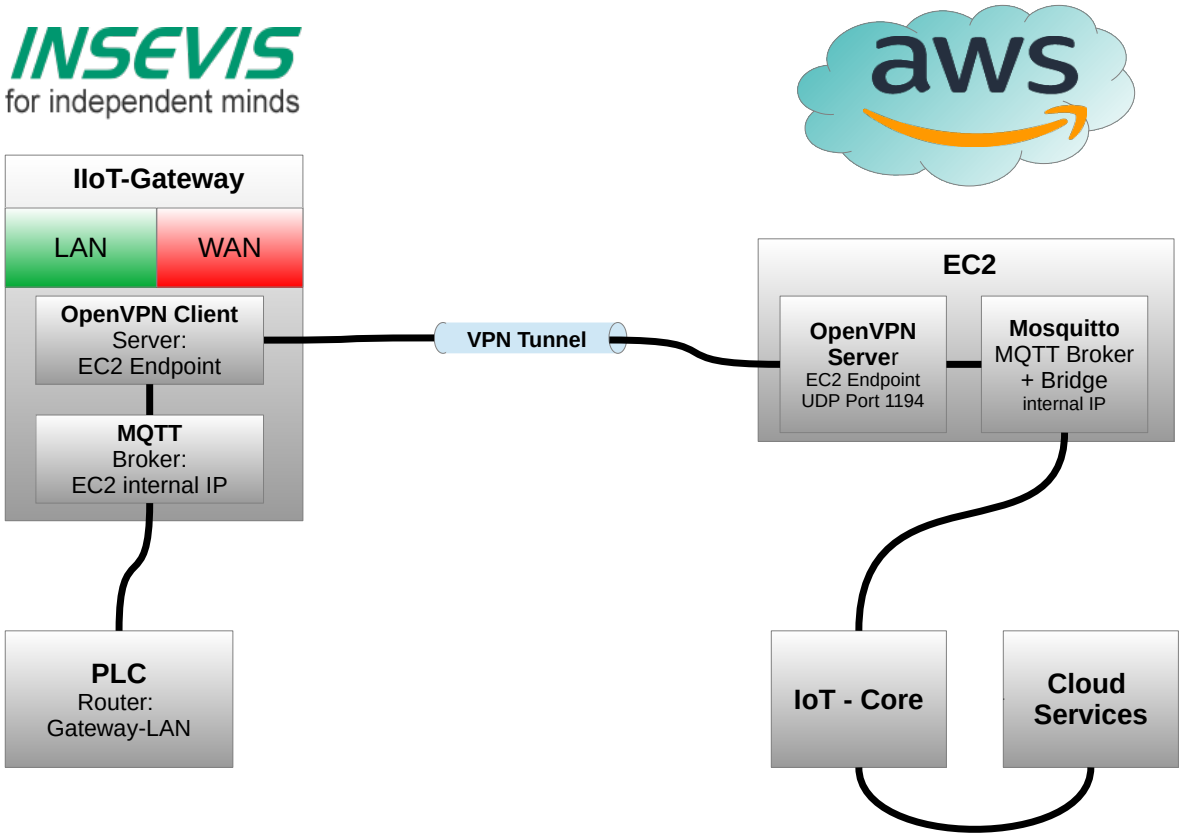
TERMS OF USE

The use of this sample programs is allowed only under acceptance of following conditions by the user:
The present software which is for guidance only aims at providing customers with sampling information regarding their S7-programs in order to save time. As a result, INSEVIS shall not be held liable for any direct, indirect or consequential damages respect to any claims arising from the content of such software and/or the use made by customers of this sampling information contained herin in connection with their own programs.

## SAMPLE DESCRIPTION AWS

**Target**

The AWS IoT Core services have security requirements for data transfer that are not yet implemented in the INSEVIS IoT gateway in the MQTT service..
A VPN tunnel into a virtual server (AWS CE2) in which an MQTT bridge forwards the data to IoT Core according to the requirements is used for secure transmission.
This VPN tunnel can be used for remote maintenance as well.




**Prerequisites and preliminary consideration**

- knowledge in the use of Linux console and INSEVIS IIoT-gateway
- an AWS account and knowledge of how to configure it,
- a PC with ssh und scp
- planning of the application (payload:  structure, names, direction)

**AWS IoT-Core**

- create a "Thing" and download an SDK

  The SDK contains the key pair (there probably are other ways)

- download root-certificat here or in EC2:
  **$ wget https://www.amazontrust.com/repository/AmazonRootCA1.pem**
  (there are also other URLs)

- setup the security policy
  (by default only the demo-topics and the demo-client-name of the SDK might be released !)
  setup topic- and connect-filter

- memorize IoT-Core Endpoint

**Configure AWS EC2**
e.g. „micro“, „debian“ „PC“
- generate and download RSA-key pair for SSH
- memorize extern IP-address (EC2-endpoint-IP)
  setup local ssh access with AWS-Key (config file .ssh containing path to private key)

→ login from PC on EC2:
**> ssh admin@***<EC2-Endpoint-IP>*

- determine local IP-address from login-promt,or per
**$ ip add**

→ memorize

- install MQTT-broker
**$ sudo apt install mosquitto**
**$ sudo apt install mosquitto-clients** (nur zu Testzwecken)

- configure MQTT-broker:
**$ sudo nano /etc/mosquitto/mosquitto.conf**
by default only "localhost" is allowed, add:
**listener 1883**

- push the 2 keys from the IoT core SDK on the PC into the EC2:
**> scp keyfile admin@***<EC2-Endpoint-IP>***:~**

- get the rootcertifiact (or - see above - with the other keys via scp)
**$ sudo wget https://www.amazontrust.com/repository/AmazonRootCA1.pem**

- move the 3 keys to /etc/mosquitto/cert (in 2 steps, because no sudo access via scp):
**$ sudo mv** *<keyfile>* **/etc/mosquitto/certs**

- configure MQTT bridge:
move „bridge.conf“ to /etc/mosquitto/conf.d/ or create and fill it:
**$ sudo nano /etc/mosquitto/conf.d/bridge.conf**

adapt it to your application:
**address** *<IoT-Core endpoint>*
**topic** *<results from the application> in 1*
**topic** *<results from the application> out 1*

restart:
**$sudo systemctl restart mosquitto**

Now you could already test the Mosquitto locally and test the bridge via the test page in the IoT core.
**$ mosquitto_sub demotopic/#**
**$ mosquitto_pub demotopic/text blabla**

- install OpenVPN
**$ sudo apt install openvpn**
(Configuration can be done only later)

**INSEVIS IIoT Gateway**

According to the planned application, a connection for the PLC and the corresponding data points must be defined. (After that the status of the connection should be "running")

- generate openVPN CA-Key,
- set extern IP of EC2 (EC2-Endpoint-IP) as "PublicIP" for openVPN-Server, (+ „save to device“)
- generate openVPN-Server:

- enable option routen server-net
create and export one or more openVPN clients
(e.g. IIoTGateway1..3, SupportPC)
route client-net if used

*Note: do NOT start openVPN server, it is NOT used here*

- make „Compete Backup“
rename <file>.bup into <file>.tgz
open and extract subdirectory "./openvpn/" (The rest can be removed.)

edit file ./openvpn/server.conf :
push "route 172.31.44.155" should contain the locale IP of EC2
(but it will contain the IP address of the gateway)
route 192.168.75.0 255.255.255.0
these line(s) allow access from the EC2 "behind" the gateway (e.g. to the PLC)

Note: This configuration of an openVPN can also be done on any Linux PC or Raspberry Pi with " standard tools". We use the web interface of the gateway here.

The configuration in the directory openVPN should now consist of /ccd, /easy-rsa and server.conf
Load this configuration from the PC into the EC2:

```
> scp -r openvpn/ admin@<EC2-Endpoint-IP>:~
```

- enable openVPN-Client:
Import back the openVPN client created for this IIoT gateway via "Import openVPN Client" from the just exported file.
- make [Start], "Start at startup" and "save to device"

→ the gateway now shows "running", but the server is still missing - but you can only see this in the logfile

IIoT-MQTT-Configuration:
- For the tunneled MQTT communication, the internal EC2 IP address is to be used as broker.
(But this does not work yet !!)
- Enter data points and assigned topics according to the defined application,
Activate "String conversion".

The EC2 endpoint (with UDP port 1883 enabled) or any other "open" test MQTT broker can be used for testing.
(With this, the status of the connection should be "running").

**AWS EC2 part 2**

Move the openVPN configuration just transferred from the PC into the system:

```
$ sudo mv ~/openvpn/ccd/         /etc/openvpn
$ sudo mv ~/openvpn/easy-rsa/    /etc/openvpn
$ sudo mv ~/openvpn/server.conf /etc/openvpn
```

start service:

```
$ sudo systemctl enable openvpn@server.service
$ sudo systemctl start  openvpn@server.service
```

In the EC2 security settings, enable UDP port 1194 for incoming traffic.

- test openVPN:

```
$ ping 10.1.0.1                          = self-ping of server
$ ping 10.1.0.2 (or .3)                  = client-ping
$ ping <IIoT-Gateway-LAN-Adresse>        = ping client
```

**Option remote maintenance**

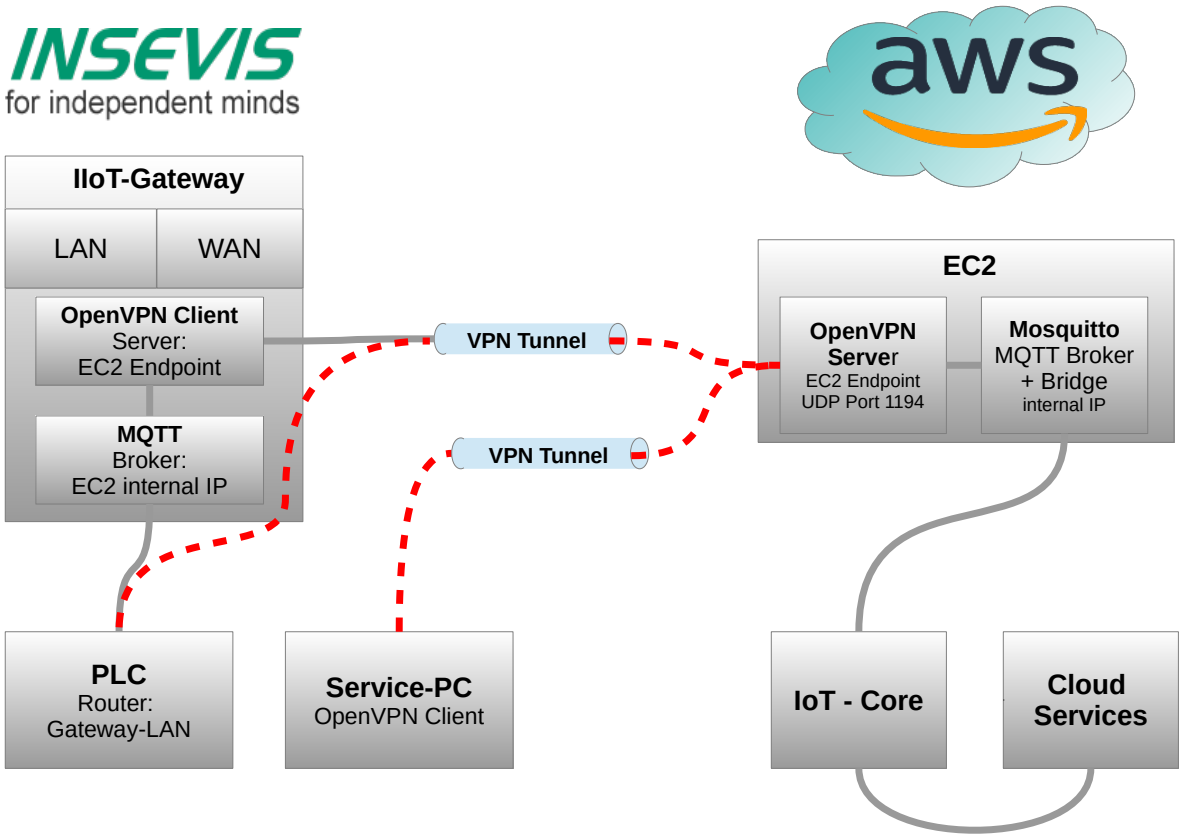
If another VPN client was generated, the ovpn file from the exported package can be used on a PC with installed openVPN client (without username/password).
The target network address (IIoT gateway LAN) must be specified to the PC. You can set the routing in the PC manually or add it to the ovpn file:

```
route 192.168.80.0 255.255.255.0
```

For a site-to-site VPN, ip-forwarding must be enabled in the EC2:

```
$ echo 1 | sudo tee -a /proc/sys/net/ipv4/ip_forward
```

Now the configuration interface of the IIoT gateway and the PLC behind it are accessible via VPN.
It is important that no IP address ranges of the local networks (IIoT gateway LAN/WAN, PC, EC2) overlap.




Since EC2 is publicly accessible as openVPN-server, no local firewall has to be configured.

**Start-up**

when S7 variables change, an MQTT message should be sent into the EC2 broker.
This can be observed - without bridge function - in the EC2 command line:

```
$ mosquitto_sub -t <mein topic>
```

If the bridge is set up correctly, the messages also appear in the AWS IoT Core MQTT test client.
There you can also send messages, which should then arrive in the EC2 as well as in the PLC.

**Hint:**

FEEDBACK
Do you want to inform us about necessary increments or errors or do you want to provide us with your sample programs to offer it for free to all customers? ***Please inform us at info@insevis.de***
Gladly we would provide your program -if you wish with the authors name- to all other customers of INSEVIS.