Infinimesh v3 Installation:

Infinimesh platform is shipped as a small local infrastructure that is running inside of a Docker engine on local computer. We provide the scripts and pre-build images with Infinimesh components that can be deployed locally inside running engine. At the end of the installation is expected to have a fully working system which later needs to be provisioned with at least one device in order to see actual communication.

Supported operating system: Any major distribution of Linux that supports Docker, MacOSX (not running on Apple Silicon), Microsoft Windows.

Who installs the platform in scope of evaluation, ideally should worked before with:

- Version control systems – git
- Docker containers
- Certificate management
- Basic understanding of networking and domain name resolution

We have included also steps for installing any missing components on local system, in case is not already available.

Check minimal requirements (common on all supported operating systems)

- git
- docker engine with compose plugin
- openssl binary and library (for generation of self-signed certificates)
- mqtt client (for testing Infinimesh platform)

For **Microsoft Windows**, install following binaries:

OpenSSL binary: https://slproweb.com/products/Win32OpenSSL.html
Docker Engine (desktop): https://docs.docker.com/desktop/install/windows-install/
Git: https://github.com/git-for-windows/git/releases/tag/v2.38.1.windows.1
Mosquitto (mqtt client):
For Windows 32bit: https://mosquitto.org/files/binary/win64/mosquitto-2.0.9-install-windows-x64.exe
For Windows 64bit: https://mosquitto.org/files/binary/win32/mosquitto-2.0.9-install-windows-x86.exe

For Linux, depending of the distribution, install:

	Ubuntu / Debian	**RPM based (Redhat / CentOS, Fedora,etc)**
git	• Start by updating the package index: sudo apt update. • Run the following command to install Git: sudo apt install git. • Verify the installation by typing the following command which will print the Git version: git --version	sudo dnf update -y sudo dnf install git -y git --version

Installation of Docker engine on Linux:

Follow official documentation at: https://docs.docker.com/desktop/install/linux-install/ for your installed Linux distribution.

(for .rpm Based distribution)

```
#[root@localhost ~]# yum install -y yum-utils
Dependencies resolved.
(…)
Installed:
  yum-utils-4.0.21-3.el8.noarch
Complete!
[root@localhost ~]# yum-config-manager \
>     --add-repo \
>     https://download.docker.com/linux/centos/docker-ce.repo
Adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
[root@localhost ~]#[root@localhost ~]# yum install docker-ce docker-ce-cli
containerd.io docker-compose-plugin
Docker CE Stable - x86_64
(…)
Installing:
 containerd.io                          x86_64          1.6.10-3.1.el8
docker-ce-stable           33 M
 docker-ce                              x86_64          3:20.10.21-3.el8
docker-ce-stable           21 M
 docker-ce-cli                          x86_64          1:20.10.21-3.el8
docker-ce-stable           30 M
 docker-compose-plugin                  x86_64          2.12.2-3.el8
docker-ce-stable           10 M
Installing dependencies:
 checkpolicy                            x86_64          2.9-1.el8
baseos                    348 k
 container-selinux                      noarch          2:2.167.0-
(…)
Installed:
  checkpolicy-2.9-1.el8.x86_64
container-selinux-2:2.167.0-1.module_el8.5.0+911+f19012f9.noarch
  containerd.io-1.6.10-3.1.el8.x86_64                                  docker-
ce-3:20.10.21-3.el8.x86_64
  docker-ce-cli-1:20.10.21-3.el8.x86_64                                docker-
ce-rootless-extras-20.10.21-3.el8.x86_64
  docker-compose-plugin-2.12.2-3.el8.x86_64                            docker-
scan-plugin-0.21.0-3.el8.x86_64
```

```
  fuse-overlayfs-1.7.1-1.module_el8.5.0+890+6b136101.x86_64        fuse3-
3.2.1-12.el8.x86_64
  fuse3-libs-3.2.1-12.el8.x86_64
libcgroup-0.41-19.el8.x86_64
  libslirp-4.4.0-1.module_el8.5.0+890+6b136101.x86_64
policycoreutils-python-utils-2.9-16.el8.noarch
  python3-audit-3.0-0.17.20191104git1c2f876.el8.x86_64             python3-
libsemanage-2.9-6.el8.x86_64
  python3-policycoreutils-2.9-16.el8.noarch                        python3-
setools-4.3.0-2.el8.x86_64
  slirp4netns-1.1.8-1.module_el8.5.0+890+6b136101.x86_64

Complete
[root@localhost ~]# systemctl start docker
[root@localhost ~]#
[root@localhost ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest:
sha256:faa03e786c97f07ef34423fcccceeec2398ec8a5759259f94d99078f264e9d7af
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

(for .**deb** Based distributions)
In this example is used Ubuntu version 22.04 LTS, server edition.

1. Check if **git** is installed. On this installation is already installed, but in case is not available, install it using the package manager.

```
root@infinimesh:~# git
-bash: /usr/bin/git: No such file or directory
```

If the git application is not installed or not available into current user path, an error will be shown as in previous message. Install it manually:

```
root@infinimesh:~# apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk
gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git
0 upgraded, 1 newly installed, 0 to remove and 48 not upgraded.
(…)
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.5_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.5) ...
Setting up git (1:2.34.1-1ubuntu1.5) ...
```

Check if openssl binary is installed:

```
root@infinimesh:/opt# openssl version
OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)
```

Install an mqtt client – in our case mosquitto. We will use it later for checking the platform and see how messages are sent / received.

```
root@infinimesh:/opt# apt install mosquitto-clients
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer
required:
  libev4 libwebsockets16
Use 'apt autoremove' to remove them.
The following NEW packages will be installed:
  mosquitto-clients
0 upgraded, 1 newly installed, 0 to remove and 48 not upgraded.
Need to get 117 kB of archives.
After this operation, 297 kB of additional disk space will be used.
(…)
Preparing to unpack .../mosquitto-clients_2.0.15a-
0mosquitto1~jammy1_amd64.deb ...
Unpacking mosquitto-clients (2.0.15a-0mosquitto1~jammy1) ...
Setting up mosquitto-clients (2.0.15a-0mosquitto1~jammy1) ...
Processing triggers for man-db (2.10.2-1) ...
```

Next step is to install Docker engine and compose plugin which will be used by Infinimesh platform in order to run on local machine.

For detailed installation of docker engine on Ubuntu see:
https://docs.docker.com/engine/install/ubuntu/

Install using the repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Set up the repository

Update the *apt* package index and install packages to allow apt to use a repository over HTTPS:

```
root@infinimesh:~# apt-get update
Hit:1 http://de.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://de.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:3 http://de.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8
kB]
(…)
Fetched 3,303 kB in 3s (1,182 kB/s)
Reading package lists... Done
```

Install dependencies:

```
root@infinimesh:~# apt-get install ca-certificates \
    curl gnupg lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20211016).
ca-certificates set to manually installed.
lsb-release is already the newest version
```

Add Docker's official GPG key:

```
root@infinimesh:~# mkdir -p /etc/apt/keyrings
root@infinimesh:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg
| sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Use the following command to set up the repository:

```
root@infinimesh:~# echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list
> /dev/null
```

Install Docker Engine

Update package list with the newly added repo. If is not updated, next command which installs Docker Engine, will not find the new packages and will throw an error.

```
root@infinimesh:~# apt update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.9 kB]
Hit:2 http://de.archive.ubuntu.com/ubuntu jammy InRelease
(…)
Get:4 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
[9,632 B]
 Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
48 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Install Docker Engine binaries:

```
root@infinimesh:~# apt-get install docker-ce docker-ce-cli  docker-compose-
plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-ce-rootless-extras docker-scan-plugin libltdl7
libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
(…)
After this operation, 428 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

```
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64
containerd.io amd64 1.6.10-1 [27.7 MB]
Get:2 http://de.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64
2.6-1 [63.6 kB]
Get:3 http://de.archive.ubuntu.com/ubuntu jammy/main amd64 libltdl7 amd64
2.4.6-15build2 [39.6 kB]
(…)
Selecting previously unselected package containerd.io.
Preparing to unpack .../1-containerd.io_1.6.10-1_amd64.deb ...
Unpacking containerd.io (1.6.10-1) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../2-docker-ce-cli_5%3a20.10.21~3-0~ubuntu-
jammy_amd64.deb ...
Unpacking docker-ce-cli (5:20.10.21~3-0~ubuntu-jammy) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../3-docker-ce_5%3a20.10.21~3-0~ubuntu-jammy_amd64.deb
...
Unpacking docker-ce (5:20.10.21~3-0~ubuntu-jammy) ...
(…)
Unpacking docker-compose-plugin (2.12.2~ubuntu-jammy) ...
(…)
Created symlink /etc/systemd/system/multi-
user.target.wants/containerd.service →
/lib/systemd/system/containerd.service.
Setting up docker-compose-plugin (2.12.2~ubuntu-jammy) ...
Setting up libltdl7:amd64 (2.4.6-15build2) ...
Setting up docker-ce-cli (5:20.10.21~3-0~ubuntu-jammy) ...
Setting up libslirp0:amd64 (4.6.1-1build1) ...
Setting up pigz (2.6-1) ...
Setting up docker-ce-rootless-extras (5:20.10.21~3-0~ubuntu-jammy) ...
Setting up slirp4netns (1.0.1-2) ...
Setting up docker-ce (5:20.10.21~3-0~ubuntu-jammy) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service
→ /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket →
/lib/systemd/system/docker.socket.
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Scanning processes...
Scanning linux images...
```

At this point Docker Engine is installed. We can verify the installation by starting a test container:

```
root@infinimesh:~# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest:
sha256:faa03e786c97f07ef34423fccceeec2398ec8a5759259f94d99078f264e9d7af
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Once is confirmed the installation of Docker Engine, we can move forward with deployment of Infinimesh platform. This steps are common on all supported operating systems.

Download of the Infinimesh platform

From command line, issue following commands:

Download Infinimesh platform:

```
[root@localhost infinimesh]#git clone
https://github.com/infinimesh/infinimesh.git
Cloning into 'infinimesh'...
remote: Enumerating objects: 35637, done.
remote: Counting objects: 100% (841/841), done.
remote: Compressing objects: 100% (314/314), done.
Receiving objects:  24% (8629/35637), 69.82 MiB | 3.46 MiB/s
```

After download has finish, access the folder and start the platform

```
[root@localhost ~]#
[root@localhost infinimesh]# ls
CODE_OF_CONDUCT.md  Dockerfiles  README.md         asciicast.gif  console
e2e     go.sum  pkg          vetur.config.js
CONTRIBUTING.md     LICENSE      api.swagger.json  cmd            docker-
compose.yaml  go.mod  hack    traefik.yml
[root@localhost infinimesh]# docker compose up -d
[+] Running 18/50
 ⠧ redis Pulling
13.6s
   ⠧ a603fa5e3b41 Waiting
6.5s
   ⠧ 77631c3ef092 Waiting
6.5s
   ⠧ ed3847cf62b8 Waiting
6.5s
   ⠧ 261a8b530567 Waiting
6.5s
   ⠧ 7d9005a8af6d Waiting
6.5s
   ⠧ 828da1afb5be Waiting
6.5s
 ⠧ db Pulling
13.6s
   ⠿ 213ec9aee27d Pull complete
(…)
[+] Running 14/14
 ⠿ Network infinimesh_default        Created
0.5s
 ⠿ Volume "infinimesh_data"          Created
0.0s
 ⠿ Volume "infinimesh_user-media"    Created
0.0s
 ⠿ Container infinimesh-http-fs-1    Started
1.4s
 ⠿ Container infinimesh-rabbitmq-1   Started
1.6s
 ⠿ Container infinimesh-handsfree-1  Started
1.6s
```

```
 ⠿ Container infinimesh-redis-1      Started
1.4s
 ⠿ Container infinimesh-proxy-1      Started
1.8s
 ⠿ Container infinimesh-db-1         Started
1.5s
 ⠿ Container infinimesh-shadow-1     Started
1.9s
 ⠿ Container infinimesh-repo-1       Started
2.6s
 ⠿ Container infinimesh-mqtt-1       Started
3.7s
 ⠿ Container infinimesh-web-1        Started
3.5s
 ⠿ Container infinimesh-console-1    Started
4.2s
```

Install CLI for Infinimesh platform:

Download the precompiled version of **inf** binary from following link: https://github.com/infinimesh/inf/releases for your operating system.

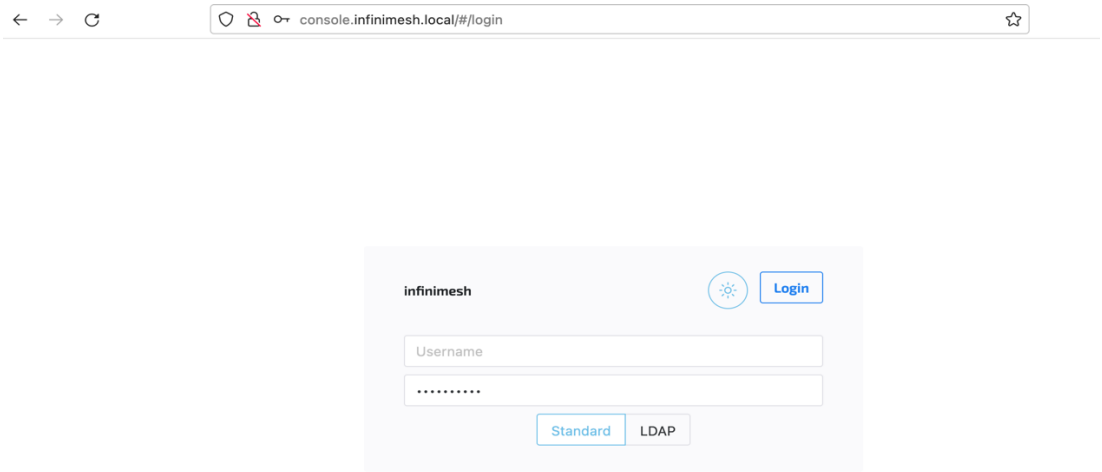
Edit hosts file (/etc/hosts) by using an editor on local operating system and add following hosts:

```
127.0.0.1 api.infinimesh.local
127.0.0.1 console.infinimesh.local
127.0.0.1 traefik.infinimesh.local
127.0.0.1 rbmq.infinimesh.local
127.0.0.1 db.infinimesh.local
127.0.0.1 media.infinimesh.local
127.0.0.1 mqtt.infinimesh.local
```

Previous step is mandatory, otherwise will not be able to access the platform.

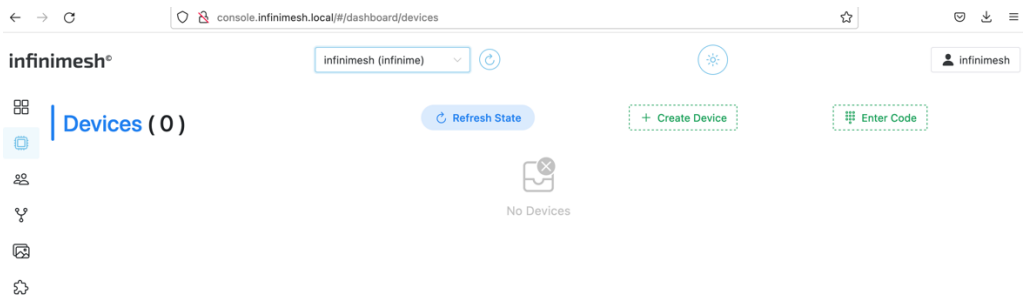
Access the console of the Infinimesh platform, by pointing your browser at the address: https://console.infinimesh.local

Use the default credentials:
Username: “infinimesh”
Password: “infinimesh”



On picture above, you can see the login page of the Infinimesh platform. Browser used on this example is Mozilla Firefox, but can be any other standard browser.

After a successful login, for the first time on a clean install, first page will look as the page bellow. In the picture bellow, can be seen than no device has been added.



Working with devices:

Infinimesh V3 supports authentication of devices using certificates pairs (public and private keys) or token based.

Please note: every device needs to have a way to authenticate.

For the first time we can use an existing certificate pair that we already have (or has been provided by us as example), but in case we want to create new ones, procedure bellow can be followed:

Authentication using certificates:

Bellow commands for certificate creation will work on Linux distributions and MacOSX. For Microsoft Windows in necessary to install openssl binary.

Generate a private key for the device:

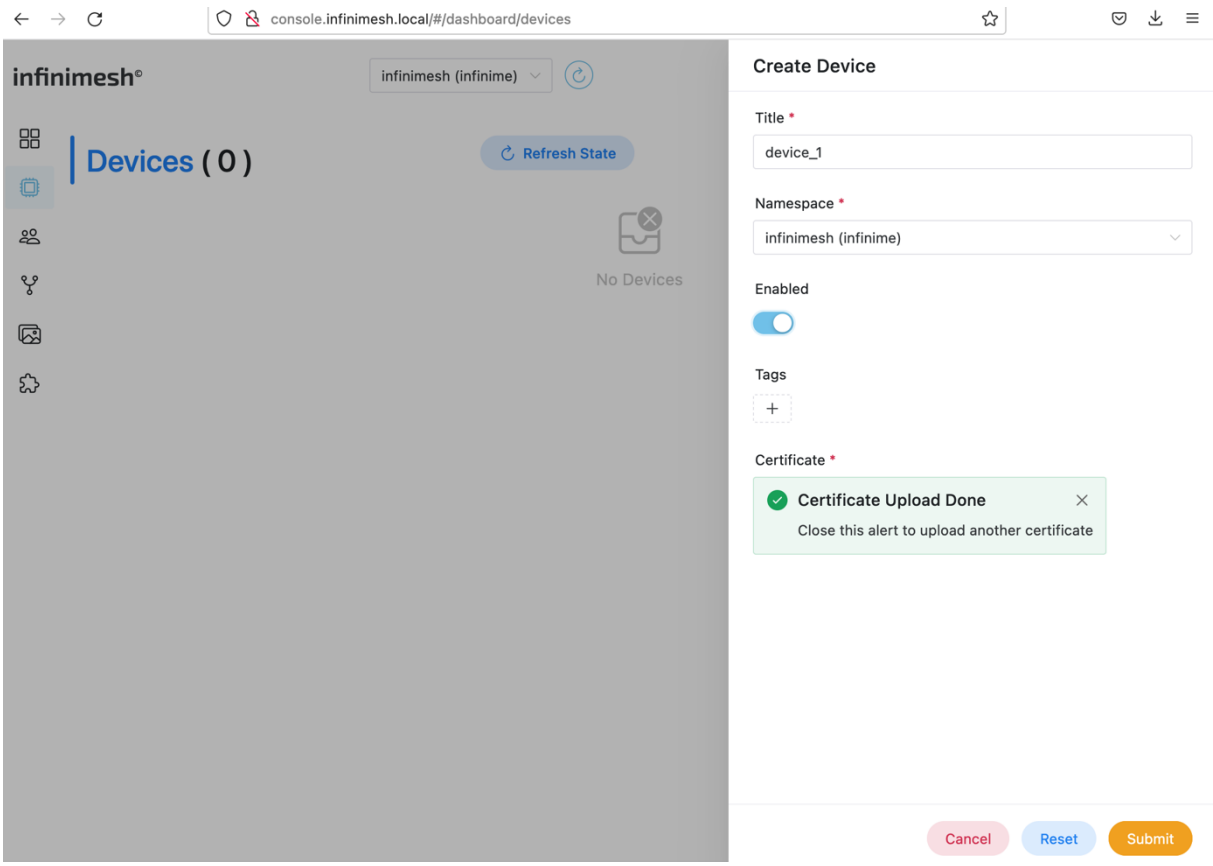
```
#openssl genrsa -out sample_1.key 4096
```

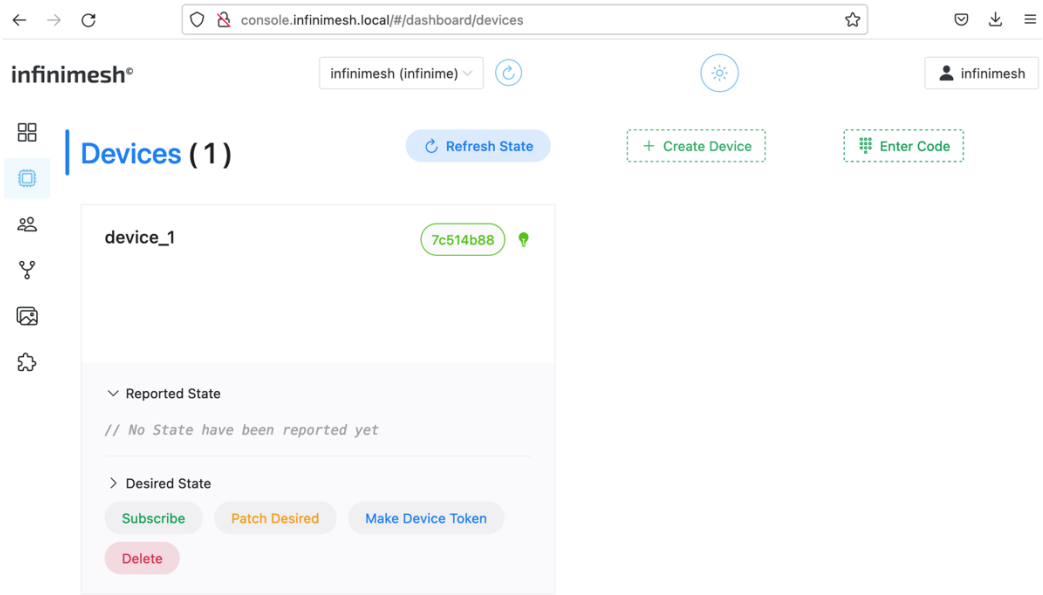
Generate the client certificate (and self-sign it):

```
#openssl req -new -x509 -sha256 -key sample_1.key -out sample_1.crt -days
365
```

Private key (sample_1.key) is on client side and the corresponding public key (sample_1.key) is uploaded into Infinimesh platform in order to validate that the connecting client is

authenticated. Device creation can be done using UI console or the cli that talks with platform through API calls.





Send states from a device to infinimesh

To simulate a device, we use the mosquitto_pub client. You can use any MQTT client, e.g. eclipse paho as well as Microsoft Edge on RaspberryPI, Yocto MQTT layers or Ubuntu Core based snaps. We use sometimes MQTTBox (http://workswithweb.com/html/mqttbox/installing_apps.html).

```
mosquitto_pub --cafile /etc/ssl/certs/ca-certificates.crt --cert
sample_1.crt --key sample_1.key -m '{"abc" : 1337}' -t "devices/<YOUR
DEVICE ID>/state/reported/delta" -h mqtt.api.infinimesh.io  --tls-version
tlsv1.2 -d -p 8883
```

Please note:

- you will most likely need to adjust the --cafile flag to the path to the CA certificates on your system. This is OS specific.
- -t "devices/<YOUR DEVICE ID>/state/reported/delta" is a placeholder and needs to be replaced with actual ID of the device.

Bellow is an actual example, with DeviceID 0x1. On your installation might be different.

```
Client mosqpub|3396-thinkpad sending CONNECT
Client mosqpub|3396-thinkpad received CONNACK (0)
Client mosqpub|3396-thinkpad sending PUBLISH (d0, q0, r0, m1,
'devices/0x1/state/reported/delta', ... (14 bytes))
Client mosqpub|3396-thinkpad sending DISCONNECT
```

The data has been sent successfully to the platform. To send more than one value per API call you can use JSON arrays in any complexity (https://www.w3schools.com/js/js_json_arrays.asp).

Read device data from the platform

We managed to send data from a device to the platform. Now let's read back the device data from infinimesh! You can do this via gRPC or HTTP API. The simplest way is with the CLI (which uses gRPC).

```
inf state get 0x8a
```

Replace `0x8a` with the ID of your device. The output will look like this:

```
Reported State:
  Version:    2
  Timestamp:  2019-03-30 20:53:41.844158131 +0100 CET
  Data:
    {
      "abc": 1337
    }
Desired State: <none>
Configuration: <none>
```

In this case, the device sent a datapoint `abc` with the value `1337` at `20:53:41`.

Send states from the platform to the device

Sending states (`desired states`) to a device is very simple. You only need to know the deviceID. Use the API, or just the CLI.

```
inf state set 0x9c 1337
```

This sends the state `1337` to the device. Note that repeatedly sending the same state does not trigger a new message every time. Only changes are sent to the device.

Once the state has been sent, you can inspect it on the server by running:

```
inf state get 0x9c
```

The state is visible in the `desired` section.