Application Note

Akademie věd České republiky
Ústav teorie informace a automatizace AV ČR, v.v.i.

Arrowhead Core System on Ubuntu 18.04

Lukas Kohout
kohoutl@utia.cas.cz

Revision history

Rev.	Date	Author	Description
0	09.18.2020	L. Kohout	Document creation – initial version
1	09.22.2020	L. Kohout	Typos, description and disclaimer
2	05.26.2021	L. Kohout	Network, Java

Contents

Acknowledgement

This work has been supported from project Arrowhead Tools, project number ECSEL 826452 and MSMT 8A19009.

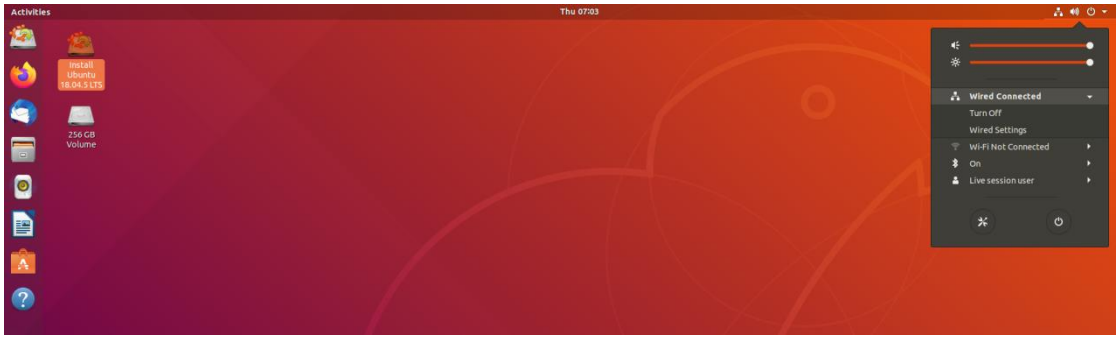
1 Description

This document describes an installation procedure of complete Arrowhead framework, running on one PC. The procedure covers installation of the operating system Ubuntu 18.04, Arrowhead system core in version 4.1.3, Arrowhead provider of the service in C++ and Arrowhead consumer of the service in C++.

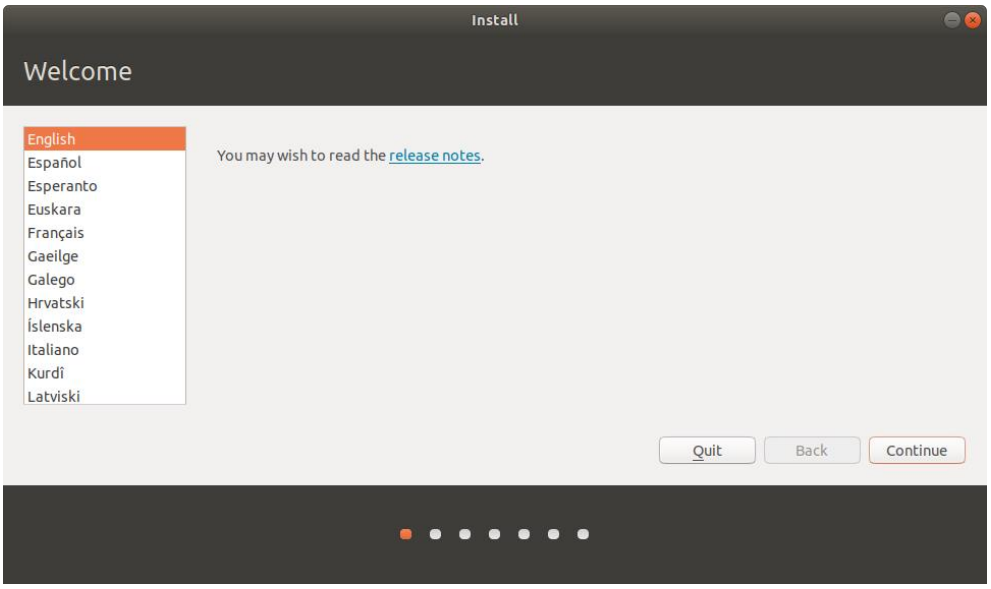
2 Ubuntu 18.04 Installation

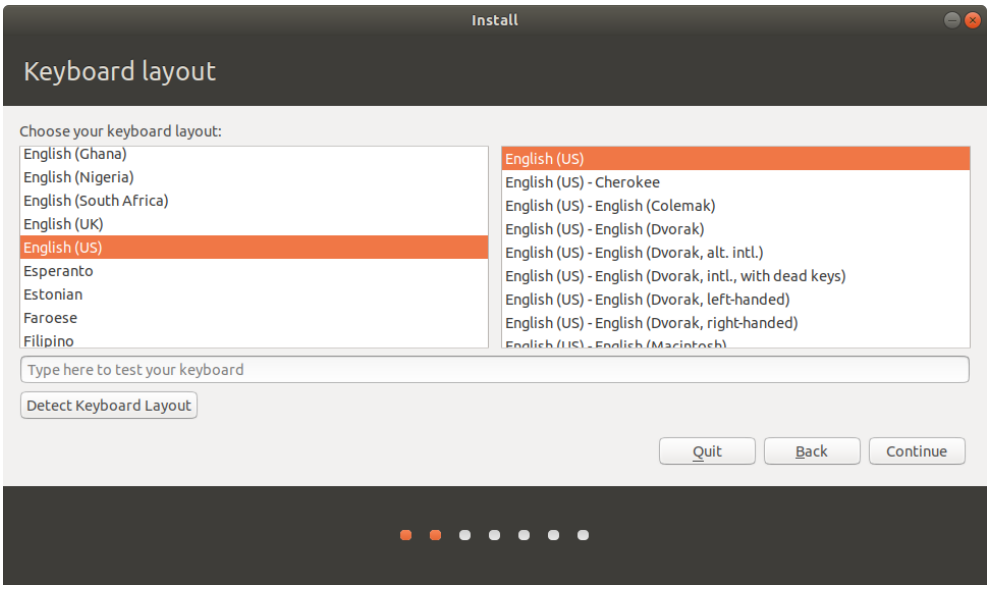
In this section there is described how to install Ubuntu 18.04 as a native operating system running on the PC. For Arrowhead Core System it is recommended to use Ubuntu 18.04 64-bit AMD64 desktop version. It is also recommended to have a PC connected to the internet. To install Ubuntu OS follow steps bellow.

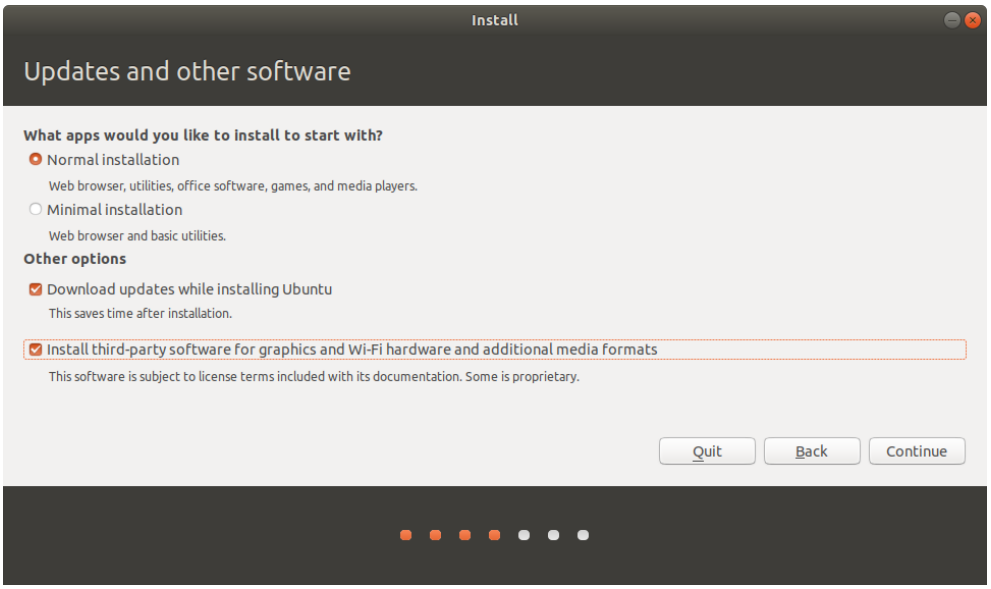
1. Download Ubuntu 18.04 64-bit AMD64 desktop ISO image:
   https://releases.ubuntu.com/18.04.5/ubuntu-18.04.5-desktop-amd64.iso
2. Create bootable installation flash drive:
   a) Form Ubuntu:
      https://ubuntu.com/tutorials/create-a-usb-stick-on-ubuntu#1-overview
   b) From Windows
      https://ubuntu.com/tutorials/create-a-usb-stick-on-windows#1-overview
3. Boot the PC from the flash, select *Try Ubuntu* if it will be offered.
4. After boot, set network connection.

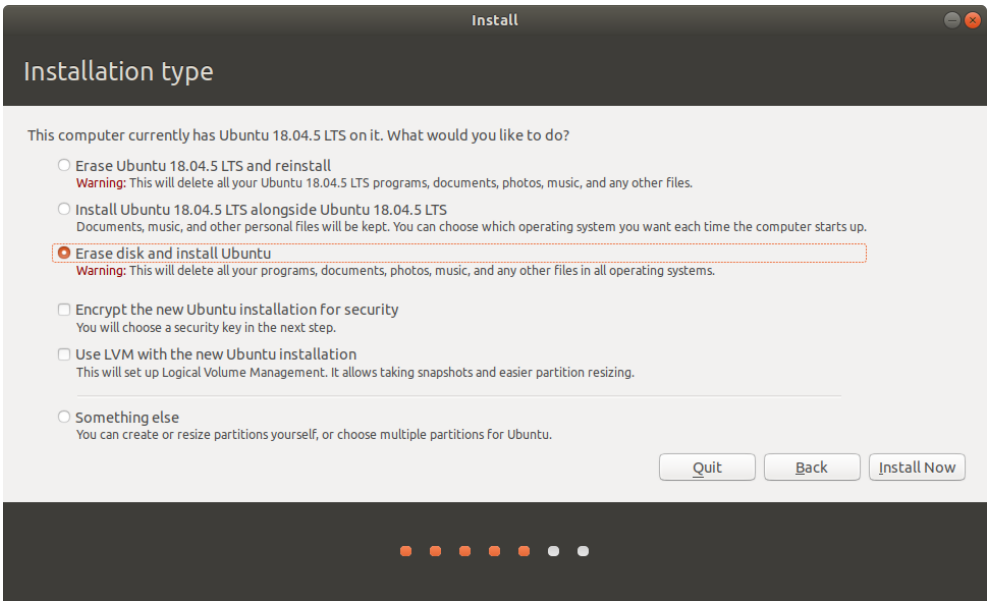


5. Install Ubuntu

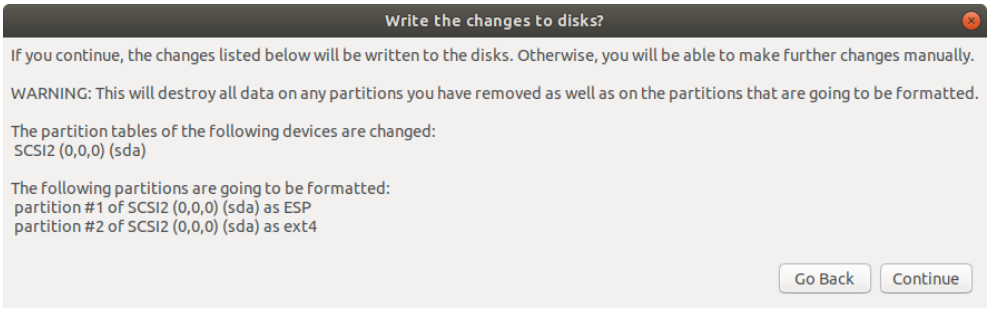


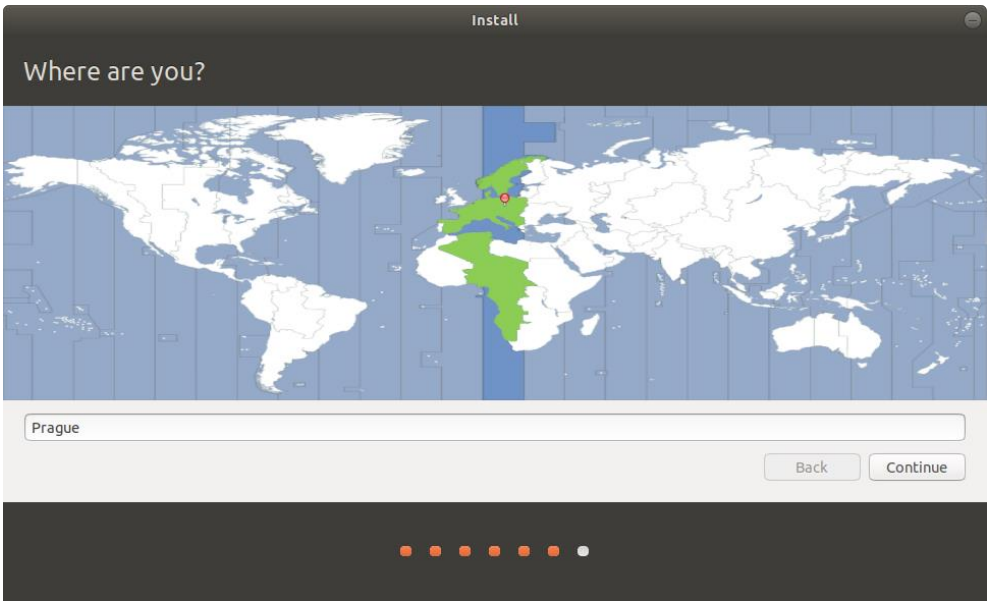


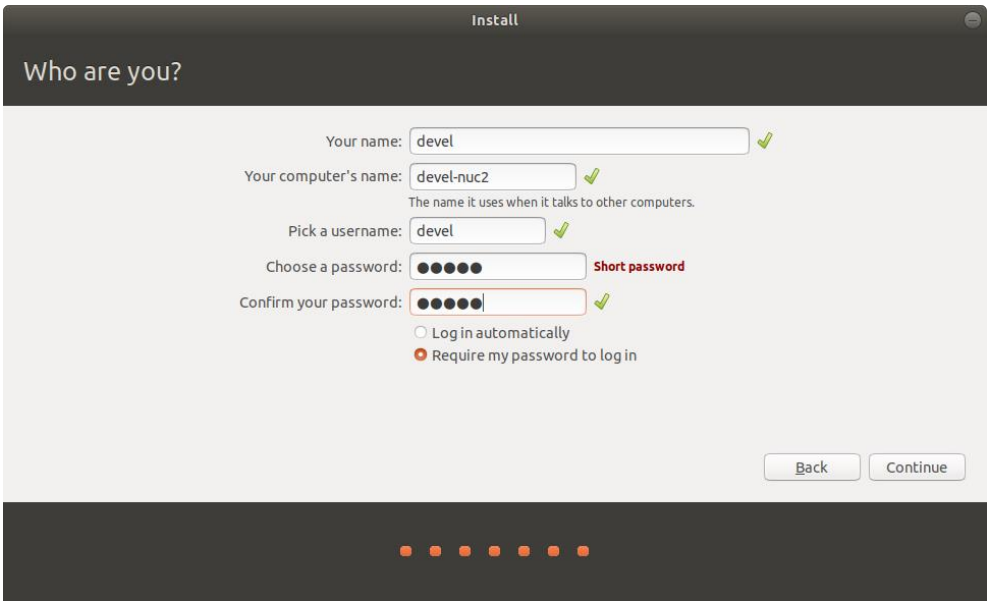




Installation type window may look differently, it depends on whether the disk is blank or there is already another OS installed on it. In our case, there is another OS and we want to reinstall it completely.







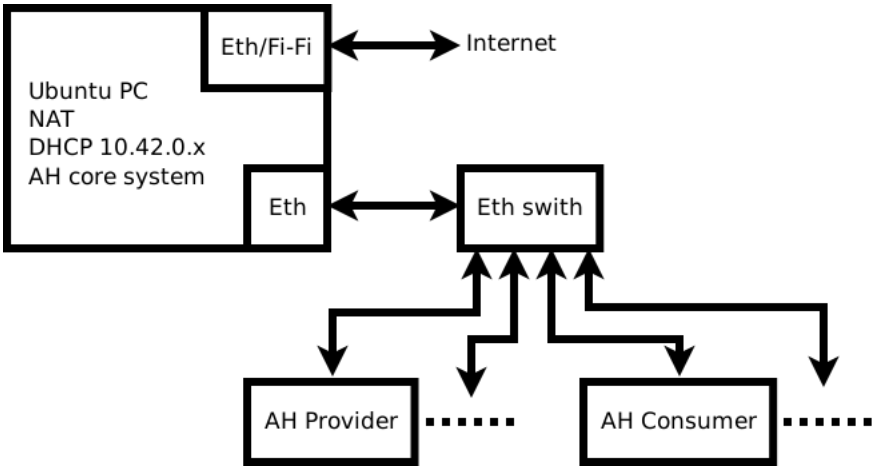
The user name *devel* and password *devel* are used in next steps of this document. If you use different name and password, modify the steps accordingly.

6. Restart to new installation of Ubuntu.

3 Post Installations and Settings

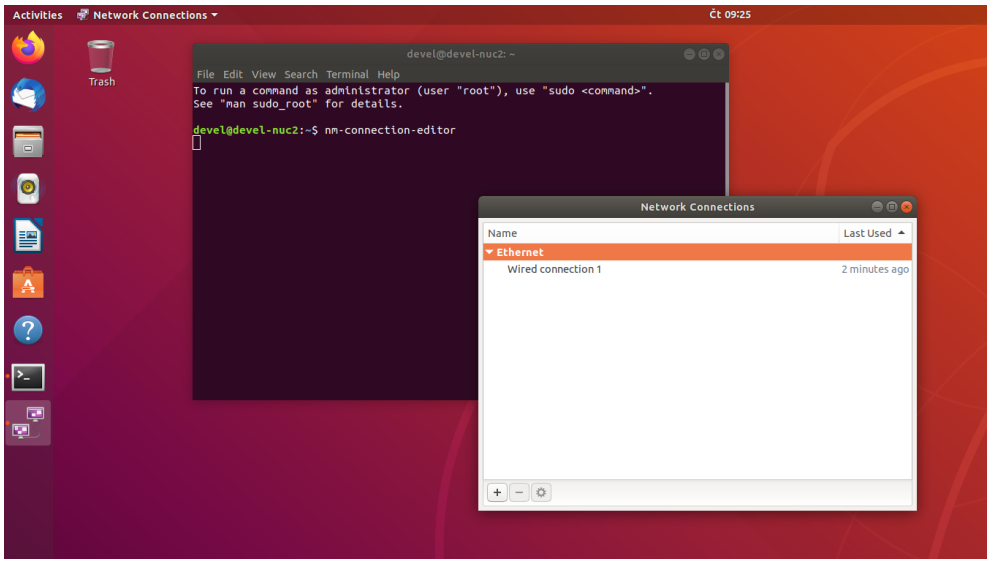
3.1 Configure internet connection

This section describes a configuration procedure of the network connection. In default configuration the Ubuntu OS is set to ask a superior DHCP server to get an IP, gateway, DNS, etc. Additionally to this configuration, another new configuration will be created. It will configure the given PC as a gateway and DHCP server and it will do NAT (Network Address Translation) for other units in the given local subnet. It is assumed that the PC has two network interfaces. The first must be Ethernet, it will be a port for the local subnet. The second interface can be any kind of and it will provide a connection to the external world. In our case the second interface is Wi-Fi adapter. The starting point is that the PC is connected via Ethernet to the local network and all described settings uses one Ethernet interface.



1. Start *nm-connection-editor*, from the terminal execute command:

```
nm-connection-editor
```

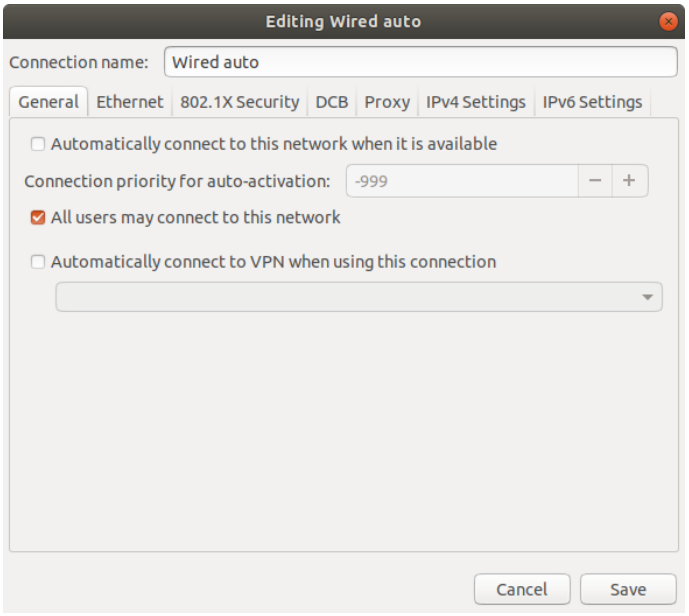


2. Modify already present network configuration that is suitable for obtaining an IP address from a superior DHCP server. This network configuration can be removed, but it is recommended to keep it to be able to switch the network configuration back to default easily. Double-click on *Wired connection 1*

a) Rename it to *Wired auto*.

b) On *General* tab deselect *Automatically connect to this network when it is available*.

c) Click *Save* button.



3. Create another connection. In this new connection, NAT will be turned on and this connection will also provide its own DHCP server. Its IP address will be 10.42.0.1 and DHCP server will provide IPs to other PCs in the same network in range 10.42.0.2 - 10.42.0.255. Click + button in *nm-connection-editor*.

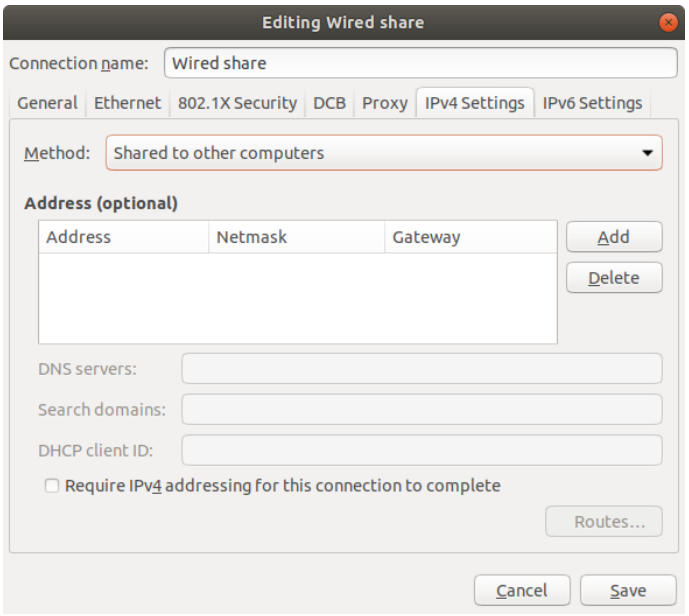
   a) Select *Ethernet* and click *Create* button.

   b) Connection name set to *Wired share*.

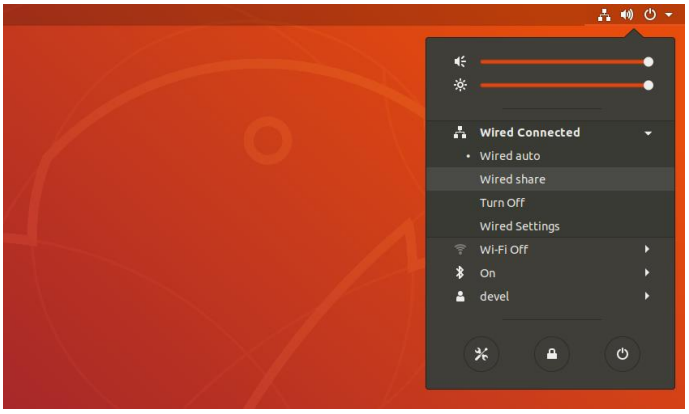
   c) On *IPv4 Settings* tab switch method to *Shared to others computers*.

   d) When you have more than one Ethernet adapter in your PC, select the interface you want to use for the local subnet. Set the proper device on *Ethernet* tab.

   e) Click *Save* button.



NOTE: It is possible to switch between these configurations. In case that the PC has two network cards (it does not matter if it is ETH or Wi-Fi) and the second one is configured to get IPs from a superior DHCP server, all computers in the local network will be able to reach the internet.



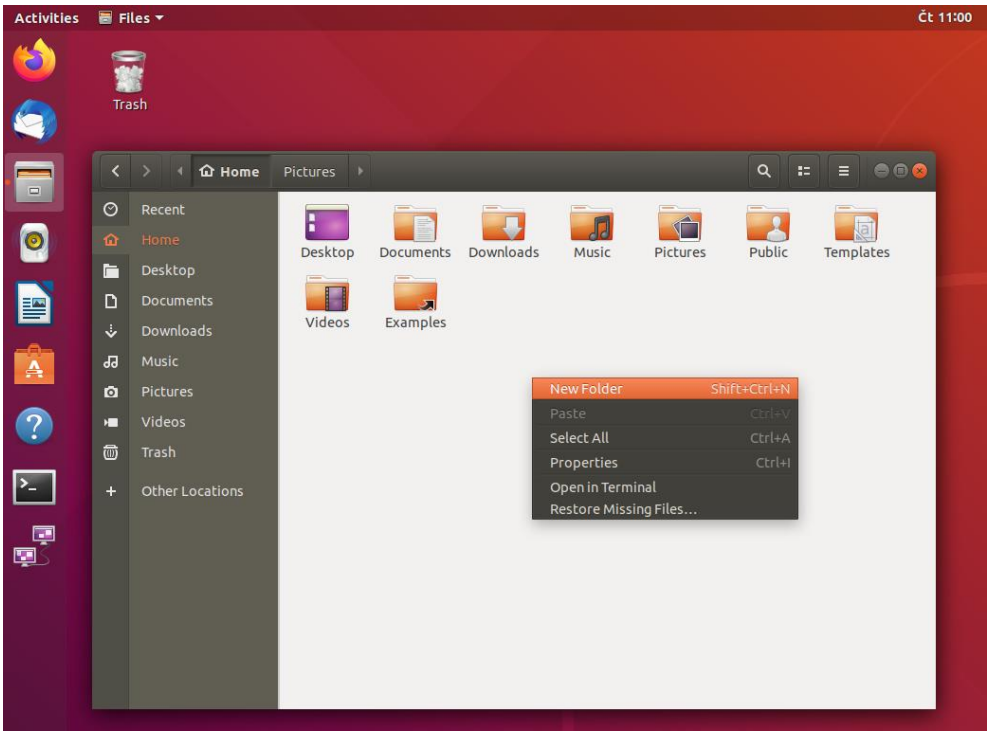
From this point of the document, all steps will require the PC connected to the internet. For the installation purpose it is possible to switch the PC back to the *Wired auto* configuration, if you have the PC connected to the internet via selected Ethernet adapter. Otherwise configure your second Ethernet port or Wi-Fi accordingly. In our case the PC has Wi-Fi adapter connected to *eduroam* Wi-Fi network.

3.2 Recommended Tools Installation

1. Install net-tools (ifconfig, etc.), from the terminal

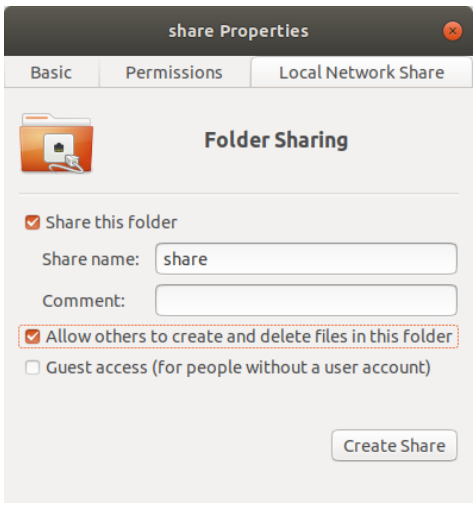
```
sudo apt install net-tools
```

2. Optionally enable user folder sharing using Samba
    a) Start file manager *nautilus*



b) In your *home* folder create folder *share*.

c) Right click on it and choose *Properties*.



d) Enable sharing of this folder. You will be probably asked to install missing services, do it.

e) Set Samba password for user *devel*. From the terminal window:

```
sudo smbpasswd -a devel.
```

Three times your password.

3. Optionally install SSH server, from the terminal:

```
sudo apt install openssh-server
```

4 Arrowhead Core System Installation

This section describes an installation procedure of the Arrowhead core system in version 4.1.3 from precompiled packages. The system has dependencies that should be installed before. They are MySQL server and Java.

4.1 MySQL Server

Arrowhead Core uses MySQL database. To install it execute from the terminal window:

```
sudo apt install mysql-server
```

4.2 Java

To install required Java runtime execute from the terminal:

```
sudo apt install openjdk-11-jre-headless
```

In case that you want to build your own packages of the Arrowhead core system from the source codes, install *openjdk11-jdk-headless* package instead. This is not our case. After system updates you can get the version of the Java that is not compatible with the Arrowhead core system we are using. The latest compatible version is 11.0.10. So check the version, from the command line:

```
java -version
```

When you get the version 10.0.10 or lower, set the system not to update the Java:

```
sudo apt-mark hold openjdk-11-jre-headless
```

When you get a higher version then 10.0.10, you have to uninstall this Java and install older version from an external source. To uninstall the Java:

```
sudo apt purge openjdk-11-jre-headless
```

Download older version:

https://builds.openlogic.com/downloadJDK/openlogic-openjdk-jre/11.0.10%2B9/openlogic-openjdk-jre-11.0.10%2B9-linux-x64-deb.deb

Go to your download folder (/home/devel/Downloads) and install downloaded Java:

```
cd ~/Downloads
sudo apt install ./openlogic-openjdk-jre-11.0.10+9-linux-x64-deb.deb
```

4.3 Arrowhead

1. Download Arrowhead system core packages, from the terminal execute commands:

```
cd ~/
mkdir ah-install
cd ah_install
wget -c https://github.com/arrowhead-f/core-java-spring-
installers/raw/master/packages/arrowhead-core-common_4.1.3.deb

wget -c https://github.com/arrowhead-f/core-java-spring-
installers/raw/master/packages/arrowhead-
authorization_4.1.3.deb

wget -c https://github.com/arrowhead-f/core-java-spring-
installers/raw/master/packages/arrowhead-
choreographer_4.1.3.deb
```

```
wget -c https://github.com/arrowhead-f/core-java-spring-
installers/raw/master/packages/arrowhead-eventhandler_4.1.3.deb

wget -c https://github.com/arrowhead-f/core-java-spring-
installers/raw/master/packages/arrowhead-gatekeeper_4.1.3.deb

wget -c https://github.com/arrowhead-f/core-java-spring-
installers/raw/master/packages/arrowhead-gateway_4.1.3.deb

wget -c https://github.com/arrowhead-f/core-java-spring-
installers/raw/master/packages/arrowhead-orchestrator_4.1.3.deb

wget -c https://github.com/arrowhead-f/core-java-spring-
installers/raw/master/packages/arrowhead-
serviceregistry_4.1.3.deb
```

2. Install arrowhead core system version 4.1.3. From the terminal execute:

```
cd ~/ah_install
sudo apt install ./arrowhead-*.deb
```

During the installation procedure you will be asked to set a couple of parameters. Follow list below. Be aware that the system is case sensitive.

a) Detached
b) testcloud
c) arrowhead
d) arrowhead
e) localhost
f) arrowhead
g) leave empty
h) arrowhead
i) arrowhead
j) arrowhead
k) arrowhead
l) arrowhead
m) arrowhead

3. Check that the installation passed successfully.

a) Check running processes of the Arrowhead core system, from the terminal:

```
ps -aux | grep arrowhead
```

Expected listing:

```
arrowhe+ 10864 30.8  7.3 5745344 562032 ?      Ssl  13:32
1:08 /usr/bin/java -
Dlog4j.configurationFile=file:/etc/arrowhead/systems/gatekee
per/log4j2.xml -jar
/usr/share/arrowhead/gatekeeper/arrowhead-gatekeeper.jar
```

```
arrowhe+ 11324 35.5  7.8 5757696 605076 ?      Ssl  13:33
1:09 /usr/bin/java -
Dlog4j.configurationFile=file:/etc/arrowhead/systems/service
_registry/log4j2.xml -jar
/usr/share/arrowhead/service_registry/arrowhead-
serviceregistry.jar
arrowhe+ 11673 40.2  6.8 5733016 525720 ?      Ssl  13:33
1:05 /usr/bin/java -
Dlog4j.configurationFile=file:/etc/arrowhead/systems/orchest
rator/log4j2.xml -jar
/usr/share/arrowhead/orchestrator/arrowhead-orchestrator.jar
arrowhe+ 11991 48.5  7.1 5744320 546456 ?      Ssl  13:34
1:02 /usr/bin/java -
Dlog4j.configurationFile=file:/etc/arrowhead/systems/event_h
andler/log4j2.xml -jar
/usr/share/arrowhead/event_handler/arrowhead-
eventhandler.jar
```

b) Check database. From the terminal execute command:

```
sudo mysql -u root
use arrowhead
show tables;
```

Expected listing:

```
+-----------------------------------------------------------+
| Tables_in_arrowhead                                       |
+-----------------------------------------------------------+
| authorization_inter_cloud                                 |
| authorization_inter_cloud_interface_connection            |
| authorization_intra_cloud                                 |
| authorization_intra_cloud_interface_connection            |
| choreographer_action                                      |
| choreographer_action_action_step_connection               |
| choreographer_action_plan                                 |
| choreographer_action_plan_action_connection               |
| choreographer_action_step                                 |
| choreographer_action_step_service_definition_connection   |
| choreographer_next_action_step                            |
| cloud                                                     |
| cloud_gatekeeper_relay                                    |
| cloud_gateway_relay                                       |
| event_type                                                |
| foreign_system                                            |
| logs                                                      |
| orchestrator_store                                        |
| relay                                                     |
| service_definition                                        |
| service_interface                                         |
| service_registry                                          |
| service_registry_interface_connection                     |
| subscription                                              |
| subscription_publisher_connection                         |
| system_                                                   |
+-----------------------------------------------------------+
```

Examine tables, *system_* for instance:

```
select * from system_;
```

Quit database:

```
quit;
```

5 Key Store Explorer Installation

To run Arrowhead clients in secure mode, they have to be equipped with its certificate that is paired with the cloud certificate. This certificate can be generated with tool called Key Store Explorer (KSE). To install the tool follow steps bellow:

1. KSE can be downloaded from https://keystore-explorer.org/downloads.html.
2. Select version for Debian based systems (*.deb). Current version of the KSE tool is kse-5.4.3.deb (2020-09-17)
3. Install KSE, from the terminal go to the download folder and install the tool:

```
cd ~/Downloads
sudo apt install ./kse-5.4.3.deb
```

4. Detailed steps describing certificate generation with the KSE tool is published on the Arrowhead core system GitHub: https://github.com/arrowhead-f/core-java-spring/blob/master/documentation/certificates/create_client_certificate.pdf.

6 Arrowhead C++ Clients Installation

This section describes an installation procedure of the Arrowhead clients coded in C++. There are two types of the client, the provider of the service and the consumer asking the service. The clients require some tools and libraries, to install them follow the steps bellow:

1. Install required tools and libraries, from the terminal execute:

```
sudo apt-get update
sudo apt-get -y upgrade

sudo apt-get -y install git openssl libgnutls28-dev \
     libgnutlsxx28 libssl1.1 libssl1.0-dev libcurl3 \
     libcurl3-gnutls libcurl4-gnutls-dev libcrypto++-dev \
     libcrypto++-utils libcrypto++6 libgpg-error-dev \
     automake texinfo g++  libjson-c-dev libjsoncpp-dev make

cd ~/
mkdir ah-client
cd ah-client

wget https://ftp.gnu.org/gnu/libmicrohttpd/libmicrohttpd-
0.9.59.tar.gz

tar -xvzf libmicrohttpd-0.9.59.tar.gz

cd libmicrohttpd-0.9.59
./configure --with-gnutls

make
sudo make install
```

```
sudo ln -sf /usr/local/lib/libmicrohttpd.so.12.46.0 \
     /usr/lib/libmicrohttpd.so.12
```

2. Download Arrowhead client source codes, from the terminal:

```
cd ~/
git clone https://github.com/arrowhead-f/client-cpp
```

6.1 Arrowhead Provider of Service

1. Generate certificate for the provider
   a) Create destination folder for the certificate, from the terminal

```
cd ~/ah-client/client-cpp/ProviderExample/
mkdir keys3
cd keys3
```

   b) Start KSE as root, from the terminal execute:

```
sudo kse
```

   c) To generate the certificate, follow steps described in:

   https://github.com/arrowhead-f/core-java-spring/blob/master/documentation/certificates/create_client_certificate.pdf

   Modify the steps in the list bellow:

   - In the 1st step of the description, the cloud certificate is located in:

```
/etc/arrowhead/clouds/testcloud.p12
```

   - In the 5th step, the *Common name (CN)* should be:

```
my_sensor.testcloud.arrowhead.arrowhead.eu
```

   - In 13th step, set alias to:

```
my_sensor
```

   - In the 15th step, save the file to the prepared folder:

```
~/ah-client/client-cpp/ProviderExample/keys3
```

     As the name of the file, use:

```
my_sensor.p12
```

   d) Convert the certificate file to set of files used by source code of the provider. From the terminal execute commands:

```
sudo chown devel:devel my_sensor.p12

openssl pkcs12 -in my_sensor.p12 -out
my_sensor.testcloud.cacert.pem -cacerts -nokeys

openssl pkcs12 -in my_sensor.p12 -out
my_sensor.testcloud.clcert.pem -clcerts -nokeys

openssl pkcs12 -in my_sensor.p12 -out
my_sensor.testcloud.privkey.pem -nocerts
```

```
openssl rsa -in my_sensor.testcloud.privkey.pem -pubout
-out my_sensor.testcloud.pubkey.pem
```

Password of the first command is *devel*, all other passwords are *arrowhead*.

2. Modify provider source code to use generated certificate, change files:
   - ~/ah-client/client-cpp/ProviderExample/src/Interface/Https_Handler.cpp.
     - On line 56 replace string

       ```
       keys2/tempsensor.testcloud2.clcert.pem
       ```

       with

       ```
       keys3/my_sensor.testcloud.clcert.pem
       ```

     - On line 63 replace string

       ```
       keys2/tempsensor.testcloud2.privkey.pem
       ```

       with

       ```
       keys3/my_sensor.testcloud.privkey.pem
       ```

     - On line 69 replace string

       ```
       123456
       ```

       with

       ```
       arrowhead
       ```

     - On line 73 replace string

       ```
       keys2/tempsensor.testcloud2.caCert.pem
       ```

       with

       ```
       keys3/my_sensor.testcloud.cacert.pem
       ```

     - On line 315 replace string

       ```
       123456
       ```

       with

       ```
       arrowhead
       ```

   - ~/ah-client/client-cpp/ProviderExample/src/Interface/Https_Handler.hpp.
     - On line 16 replace string

       ```
       keys2/tempsensor.testcloud2.privkey.pem
       ```

       with

       ```
       keys3/my_sensor.testcloud.privkey.pem
       ```

     - On line 17 replace string

       ```
       keys2/tempsensor.testcloud2.clcert.pem
       ```

       with

       ```
       keys3/my_sensor.testcloud.clcert.pem
       ```

- On line 18 replace string

  `keys2/tempsensor.testcloud2.cacert.pem`

  with

  `keys3/my_sensor.testcloud.cacert.pem`

- ~/ah-client/client-cpp/ProviderExample/src/Provider/ProvidedService.h.
  - On line 11 replace string

    `SecureTemperatureSensor`

    with

    `my_sensor`
  - On line 12 replace string

    `IndoorTemperature_ProviderExample`

    with

    `my_sensor_example`
  - On line 14 replace string

    `keys2/tempsensor.testcloud2.privkey.pem`

    with

    `keys3/my_sensor.testcloud.privkey.pem`
  - On line 15 replace string

    `keys2/tempsensor.testcloud2.pubkey.pem`

    with

    `keys3/my_sensor.testcloud.pubkey.pem`

3. Compile the provider, from the terminal execute commands:

```
cd ~/ah-client/client-cpp/ProviderExample/
make clean
make
```

4. Configure the provider to run on the same machine as the Arrowhead core system, the location and name of the configuration file:

```
~/ah-client/client-cpp/ProviderExample/
ApplicationServiceInterface.ini
```

   The configuration file consists of the following items.

   - sr_base_uri – an address of the Arrowhead registration service running in insecure mode and corresponding port.
   - sr_base_uri_https – an address of the Arrowhead registration service running in secure mode and corresponding port.
   - port – a port number where the Provider will be available on, set 8000. In this case, port 8000 is dedicated for insecure mode, for secure mode the provider automatically select port 8001.
   - address – Provider IP address.

- Address6 - Provider IP address in IPV6

The configuration file example for provider running on the same machine as the Arrowhead core system:

```
[Server]
sr_base_uri="http://10.42.0.1:8443/serviceregistry/"
sr_base_uri_https="https://10.42.0.1:8443/serviceregistry/"
port="8000"
address="10.42.0.1"
address6="[fe80::d5b:eef5:8b42:6e16]"
```

5. Start the provider, from the terminal execute:

```
./ProviderExample --secureArrowheadInterface
--secureProviderInterface
```

The provider registers itself in the arrowhead database, to check it follow the steps in Section 4.3 step 3 – b. The listing of successfully started provider:

```
============================
Provider Example - v4.1.3
============================

----------------------------
ProvidedService
----------------------------
Custom URL : /this_is_the_custom_url
System name : my_sensor
Service definition : my_sensor_example
Service interface : HTTP-SECURE-JSON
Private key path : keys3/my_sensor.testcloud.privkey.pem
Public key path : keys3/my_sensor.testcloud.pubkey.pem
Meta values:

(HTTP Server) started - 10.42.0.1:8000
(HTTPs Server) started - 10.42.0.1:8001

Measured value received from: (Base Name: this_is_the_sensor_id)
Provider is not registered yet!

REGISTRATION (Secure Provider, Secure AHInterface)

pubkeyContent:
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt+GxGuV7Stnrom1arkNEYOZ8nV5OUiIUY
P61aJ5CHVwqC+lzQQAinzxtBVP/xtfg1Zyg7wALRtvc2tjU9sZcUnUF8sRRol+6x1lbXfuYUmH/Ci
OONrrOOfgs/q6zR+cKsA+iQJ06zGQ2bQuTiD9On8AOTxAIkJXeoZ+vcWSLTa9qKrs9TzBOYN1+Wp5
zOoIvlRkpPaCb6JkE+vuhBB/kt7rlIPMKWITZRh+rTloi/g+Fvbcb8WHb61KPAAKEbt6joM9SjVbs
1mYI+WqufLq7nn9QstkkgFUT+CyBqWOvpxJeBeD5joqXVlqI6n+wnt4ZVjs9CPYtKFNTnJWZbkLhz
QIDAQAB


{ "serviceDefinition": "my_sensor_example", "serviceUri":
"\/this_is_the_custom_url", "version": 1, "secure": "TOKEN",
"providerSystem": { "systemName": "my_sensor", "address": "10.42.0.1",
"authenticationInfo":
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt+GxGuV7Stnrom1arkNEYOZ8nV5OUiIU
YP61aJ5CHVwqC+lzQQAinzxtBVP\/xtfg1Zyg7wALRtvc2tjU9sZcUnUF8sRRol+6x1lbXfuYUmH\
/CiOONrrOOfgs\/q6zR+cKsA+iQJ06zGQ2bQuTiD9On8AOTxAIkJXeoZ+vcWSLTa9qKrs9TzBOYN1
+Wp5zOoIvlRkpPaCb6JkE+vuhBB\/kt7rlIPMKWITZRh+rTloi\/g+Fvbcb8WHb61KPAAKEbt6joM
9SjVbs1mYI+WqufLq7nn9QstkkgFUT+CyBqWOvpxJeBeD5joqXVlqI6n+wnt4ZVjs9CPYtKFNTnJW
ZbkLhzQIDAQAB", "port": 8001 }, "interfaces": [ "HTTP-SECURE-JSON" ],
"metadata": { "unit": "Celsius", "security": "token" } }
SendHttpsRequest: https://10.42.0.1:8443/serviceregistry/register
HTTPs Post sent (SenML baseName = this_is_the_sensor_id)
```

```
HTTPs Post return value: 400
Already registered?
Try re-registration
SendHttpsRequest:
https://10.42.0.1:8443/serviceregistry/unregister?service_definition=my_senso
r_example&system_name=my_sensor&address=10.42.0.1&port=8001
Unregistration is successful

{ "serviceDefinition": "my_sensor_example", "serviceUri":
"\/this_is_the_custom_url", "version": 1, "secure": "TOKEN",
"providerSystem": { "systemName": "my_sensor", "address": "10.42.0.1",
"authenticationInfo":
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt+GxGuV7Stnrom1arkNEYOZ8nV5OUiIU
YP61aJ5CHVwqC+lzQQAinzxtBVP\/xtfg1Zyg7wALRtvc2tjU9sZcUnUF8sRRol+6x1lbXfuYUmH\
/CiOONrrOOfgs\/q6zR+cKsA+iQJ06zGQ2bQuTiD9On8AOTxAIkJXeoZ+vcWSLTa9qKrs9TzBOYN1
+Wp5zOoIvlRkpPaCb6JkE+vuhBB\/kt7rlIPMKWITZRh+rTloi\/g+Fvbcb8WHb61KPAAKEbt6joM
9SjVbs1mYI+WqufLq7nn9QstkkgFUT+CyBqWOvpxJeBeD5joqXVlqI6n+wnt4ZVjs9CPYtKFNTnJW
ZbkLhzQIDAQAB", "port": 8001 }, "interfaces": [ "HTTP-SECURE-JSON" ],
"metadata": { "unit": "Celsius", "security": "token" } }
SendHttpsRequest: https://10.42.0.1:8443/serviceregistry/register
Provider Registration is successful!
```

6.2 Arrowhead Consumer of Service

1. Generate certificate for the consumer
   a) Create destination folder for the certificate, from the terminal

```
cd ~/ah-client/client-cpp/ConsumerExample/
mkdir keys3
cd keys3
```

   b) Start KSE as root, from the terminal execute:

```
sudo kse
```

   c) To generate the certificate, follow steps described in:

   https://github.com/arrowhead-f/core-java-spring/blob/master/documentation/certificates/create_client_certificate.pdf

   Modify the steps in the list bellow:

   - In the first step of the description, the cloud certificate is located in:

```
/etc/arrowhead/clouds/testcloud.p12
```

   - In the 5th step, the *Common name (CN)* should be:

```
my_client.testcloud.arrowhead.arrowhead.eu
```

   - In 13th step, set alias to:

```
my_client
```

   - In the 15th step, save the file to the prepared folder:

```
~/ah-client/client-cpp/ConsumerExample/keys3
```

     As the name of the file, use:

```
my_client.p12
```

d) Convert the certificate file to set of files used by source code of the consumer. From the terminal execute commands:

```
sudo chown devel:devel my_client.p12

openssl pkcs12 -in my_client.p12 -out
my_client.testcloud.cacert.pem -cacerts -nokeys

openssl pkcs12 -in my_client.p12 -out
my_client.testcloud.clcert.pem -clcerts -nokeys

openssl pkcs12 -in my_client.p12 -out
my_client.testcloud.privkey.pem -nocerts

openssl rsa -in my_client.testcloud.privkey.pem -pubout
-out my_client.testcloud.pubkey.pem
```

Password of the first command is *devel*, all other passwords are *arrowhead*.

2. Modify consumer source code to use generated certificate, change files:
   - ~/ah-client/client-cpp/ConsumerExample/src/Consumer/ConsumerExample.cpp
     - On line 47 replace string

       ```
       client1
       ```

       with

       ```
       my_client
       ```
   - ~/ah-client/client-cpp/ConsumerExample/src/Consumer/SensorHandler.cpp
     - On line 252 replace string

       ```
       keys2/clcert.pem
       ```

       with

       ```
       keys3/my_client.testcloud.clcert.pem
       ```
     - On line 258 replace string

       ```
       keys2/privkey.pem
       ```

       with

       ```
       keys3/my_client.testcloud.privkey.pem
       ```
     - On line 264 replace string

       ```
       123456
       ```

       with

       ```
       arrowhead
       ```
     - On line 267 replace string

       ```
       keys2/cacert.pem
       ```

       with

       ```
       keys3/my_client.testcloud.cacert.pem
       ```

- ~/ah-client/client-cpp/ConsumerExample/src/Interface/Https_Handler.cpp
  - On line 56 replace string

    ```
    keys2/clcert.pem
    ```

    with

    ```
    keys3/my_client.testcloud.clcert.pem
    ```

  a) On line 62 replace string

    ```
    keys2/privkey.pem
    ```

    with

    ```
    keys3/my_client.testcloud.privkey.pem
    ```

  b) On line 68 replace string

    ```
    123456
    ```

    with

    ```
    arrowhead
    ```

  c) On line 71 replace string

    ```
    keys2/cacert.pem
    ```

    with

    ```
    keys3/my_client.testcloud.cacert.pem
    ```

  d) On line 230 replace string

    ```
    123456
    ```

    with

    ```
    arrowhead
    ```

3. Compile the consumer, from the terminal execute commands:

```
cd ~/ah-client/client-cpp/ConsumerExample
make clean
make
```

4. Configure the consumer to run on the same machine as the Arrowhead core system, the location and name of the configuration file :

```
~/ah-client/client-cpp/ConsumerExample/
OrchestratorInterface.ini
```

The configuration file consists of the following items.

- or_base_uri – an address of the Arrowhead orchestrator service running in insecure mode and corresponding port.
- or_base_uri_https – an address of the Arrowhead orchestrator service running in secure mode and corresponding port.
- port – a port number where the Provider will be available on, set 8002. In this case, port 8002 is dedicated for insecure mode, for secure mode the consumer automatically select port 8003.
- address – Provider IP address.

- Address6 - Provider IP address in IPV6

The configuration file example for consumer running on the same machine as the Arrowhead core system:

```
[Server]
or_base_uri="http://10.42.0.1:8441/orchestrator/orchestration"
or_base_uri_https="https://10.42.0.1:8441/orchestrator/orchestration"
port="8002"
address="10.42.0.1"
address6="[fe80::d5b:eef5:8b42:6e16]"
```

5. Configure a service that will be asked by the consumer. Modify file:

```
~/ah-client/client-cpp/ConsumerExample/consumedServices.json
```

Appropriately modified file is can be seen in listing bellow:

```
{
      "consumerID": "my_client",
      "requestForm": {
            "orchestrationFlags": {
                  "overrideStore": true
            },
            "requestedService": {
                  "interfaceRequirements": [
                        "HTTP-SECURE-JSON"
                  ],
                  "securityRequirements": [
                        "TOKEN"
                  ],
                  "serviceDefinitionRequirement":
                        "my_sensor_example"
            },
            "requesterSystem":
            {
                  "address": "10.42.0.1",
                  "port": 8003,
                  "systemName": "my_client"
            }
      }
}
```

6. Register the consumer in the Arrowhead database. From the terminal execute:

```
sudo mysql -u root

use arrowhead
describe system_;
```

```
+--------------------+--------------+------+-----+-------------------+-----------------------------+
| Field              | Type         | Null | Key | Default           | Extra                       |
+--------------------+--------------+------+-----+-------------------+-----------------------------+
| id                 | bigint(20)   | NO   | PRI | NULL              | auto_increment              |
| system_name        | varchar(255) | NO   | MUL | NULL              |                             |
| address            | varchar(255) | NO   |     | NULL              |                             |
| port               | int(11)      | NO   |     | NULL              |                             |
| authentication_info| varchar(2047)| YES  |     | NULL              |                             |
| created_at         | timestamp    | NO   |     | CURRENT_TIMESTAMP |                             |
| updated_at         | timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+--------------------+--------------+------+-----+-------------------+-----------------------------+
```

Insert description of the consumer to the *system_* table. Prepare a public key from file

```
~/ah-client/client-
cpp/ConsumerExample/keys3/my_sensor.testcloud.pubkey.pem
```

as one line string. To insert the consumer to the table, execute:

```
insert into system_ values
(NULL,"my_client","10.42.0.1",8003,"Public key in one
line",CURRENT_TIMESTAMP,CURRENT_TIMESTAMP);
```

Example:

```
insert into system_ values
(NULL,"my_client","10.42.0.1",8003,"MIIBIjANBgkqhkiG9w0BAQEFAAO
CAQ8AMIIBCgKCAQEAl91gte7nVmrgnO69UYt6urQI5xf/r1dMkwzHl9nszfInT3
bkFtC9L376W06TpW6Metr1kzXtXtxd59YhmKDXrOxWt9mqG4CPgso83LOX9uNM3
3rJ+ElnZMoW3ztNrlkLIFO0/LvVC2liVJchCYF0D0THXSyhhGMpzjS1+HhRWLLP
Y/peNyEXRkirhjsCuJCtd4OoA1HK6CUmXEpNaaHoN5M36YQnQicRNjUGhtS80P6
m8zMyOdWfxkmR0PcOBKqgxozrdHKUwKTC/nUpu/k1ricz9FEZwRk0xU6eyFVswS
tP2RE90Kqs7VZIknHdd9UgOgpdpKjpOt9ORmsp73gtiwIDAQAB",CURRENT_TIM
ESTAMP,CURRENT_TIMESTAMP);
```

7. Link the consumer with the provider and its service.

   a) Get ID of the provider and the consumer

   ```
   select * from system_;
   ```

   From the listing of the *system_* table get IDs, locate lines with names *my_sensor* and *my_client*. It should be 8 for provider and 10 for consumer.

   b) Get ID of the provider service

   ```
   select * from service_registry;
   ```

   In the listing of the *service_registry* table locate line with the *system_id* = 8 and get corresponding *service_id*. The service ID should be 16.

   c) Link provider, its service and consumer together.

   ```
   describe authorization_intra_cloud;
   ```

   ```
   +-------------------+-----------+------+-----+-------------------+-----------------------------+
   | Field             | Type      | Null | Key | Default           | Extra                       |
   +-------------------+-----------+------+-----+-------------------+-----------------------------+
   | id                | bigint(20) | NO   | PRI | NULL              | auto_increment              |
   | created_at        | timestamp  | NO   |     | CURRENT_TIMESTAMP |                             |
   | updated_at        | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
   | consumer_system_id | bigint(20) | NO   | MUL | NULL              |                             |
   | provider_system_id | bigint(20) | NO   | MUL | NULL              |                             |
   | service_id        | bigint(20) | NO   | MUL | NULL              |                             |
   +-------------------+-----------+------+-----+-------------------+-----------------------------+
   ```

   ```
   Insert into authorization_intra_cloud values
   (NULL,CURRENT_TIMESTAMP,CURRENT_TIMESTAMP,10,8,16);
   ```

   d) Get ID of secure connection:

   ```
   select * from service_interface;
   ```

   ```
   +----+--------------------+---------------------+---------------------+
   | id | interface name     | created at          | updated at          |
   +----+--------------------+---------------------+---------------------+
   |  1 | HTTP-SECURE-JSON   | 2020-09-17 13:29:38 | 2020-09-17 13:29:38 |
   |  2 | HTTP-INSECURE-JSON | 2020-09-17 13:29:38 | 2020-09-17 13:29:38 |
   +----+--------------------+---------------------+---------------------+
   ```

e) Set connection interface to use secure mode.

```
describe authorization_intra_cloud_interface_connection;
```

```
+-----------------------------+-----------+------+-----+-------------------+-----------------------------+
| Field                       | Type      | Null | Key | Default           | Extra                       |
+-----------------------------+-----------+------+-----+-------------------+-----------------------------+
| id                          | bigint(20) | NO   | PRI | NULL              | auto_increment              |
| authorization_intra_cloud_id | bigint(20) | NO   | MUL | NULL              |                             |
| interface_id                | bigint(20) | NO   | MUL | NULL              |                             |
| created_at                  | timestamp  | NO   |     | CURRENT_TIMESTAMP |                             |
| updated_at                  | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-----------------------------+-----------+------+-----+-------------------+-----------------------------+
```

```
insert into
authorization_intra_cloud_interface_connection values
(NULL,1,1,CURRENT_TIMESTAMP,CURRENT_TIMESTAMP);

quit;
```

8. Start the consumer, from the terminal execute:

```
./ConsumerExample --secureArrowheadInterface --
secureProviderInterface
```

Consumer prints response from the provider (fake temperature 26°):

```
Provider Response:
{"e":[{"n": "this_is_the_sensor_id","v":26.0,"t":
"1600689629"}],"bn": "this_is_the_sensor_id","bu": "Celsius"}
```

On the provider side, you should see a reaction on consumer response:

```
HTTPs GET request received
Received URL: /this_is_the_custom_url
Response:
{"e":[{"n": "this_is_the_sensor_id","v":26.0,"t":
"1600689629"}],"bn": "this_is_the_sensor_id","bu": "Celsius"}
```

Disclaimer

This disclaimer is not a license and does not grant any rights to the materials distributed herewith. Except as otherwise provided in a valid license issued to you by UTIA AV CR v.v.i., and to the maximum extent permitted by applicable law:

(1) THIS APPLICATION NOTE AND RELATED MATERIALS LISTED IN THIS PACKAGE CONTENT ARE MADE AVAILABLE "AS IS" AND WITH ALL FAULTS, AND UTIA AV CR V.V.I. HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and

(2) UTIA AV CR v.v.i. shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under or in connection with these materials, including for any direct, or any indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or UTIA AV CR v.v.i. had been advised of the possibility of the same.

Critical Applications:
UTIA AV CR v.v.i. products are not designed or intended to be fail-safe, or for use in any application requiring fail-safe performance, such as life-support or safety devices or systems, Class III medical devices, nuclear facilities, applications related to the deployment of airbags, or any other applications that could lead to death, personal injury, or severe property or environmental damage (individually and collectively, "Critical Applications"). Customer assumes the sole risk and liability of any use of UTIA AV CR v.v.i. products in Critical Applications, subject only to applicable laws and regulations governing limitations on product liability.