TURKEY
ATATÜRK UNIVERSITY
COMPUTER ENGINEERING

EDGEX FOUNDRY

Emre GÜNEŞ

Turkey/Erzurum

January , 2021

FOREWORD

In this thesis, where I describe how the EdgeX platform was installed, used, its capabilities and how other modules (such as IoT) were integrated into this structure, overcoming major obstacles, I would like to express my gratitude to Bilal USANMAZ, the instructor at the Computer Engineering department at Atatürk University who gave me this task and helped me. Despite spending a month only to install the EdgeX platform, I still continued to learn and overcome this obstacle. Although a small mistake made during the installation was expensive in terms of time, after the installation phase, I accelerated my work and moved on to other stages.

I would also like to express my gratitude to my family who supported me through the problems I encountered during this installation phase. I hope that this thesis, which I have made with the widespread use of IoT, will bring me great benefits in the field. Although EdgeX Foundry is not well known in Turkey, as the concept of "Internet of Things" becomes widespread, people will inevitably have to learn about this platform. The main reason for this is that EdgeX Foundry offers excellent management and monitoring capabilities for IoT devices. In the installation and integration phases with other modules of this platform, which has a large number of functions, we will use the Linux (Ubuntu 18.04) operating system. For questions, suggestions, and complaints, you can reach me at mailto:emre_gunes1288@outlook.com

CONTENTS

Entrance

The goal of my graduation project is to demonstrate the installation, usage, capabilities, and integration of other edge platforms and IoT modules into the EdgeX Foundry platform at a basic level. EdgeX Foundry provides a user-friendly interface for managing and monitoring IoT devices and edge platforms, which are becoming increasingly widespread worldwide, and offers great convenience for people using IoT and other edge platforms. We will use the Linux (VM) environment for the platform (EdgeX) we will work on. Then, we will install the EdgeX Foundry platform on our Linux environment using commands written on the terminal screen and try to perform operations on the device through specific ports. The ports must be active for these operations to be performed, and if the installation is not done correctly, we will not have access to these ports and therefore will not be able to reach the desired interfaces.

THEORETICAL BASICS

The EdgeX Foundry has many interfaces available in the Linux (version 18.04) environment we use, some people connect through browsers (such as Google Chrome or Firefox) to port 4000 and perform operations on EdgeX's own interface, "Golang" interface. I, on the other hand, performed device operations through "Postman". Postman provides a simple, user-friendly and savable interface. We also have the chance to perform the same operations on the Golang interface, but Postman is simply a preference.

MATERIEL AND METHOD

The goal of my graduation project is to demonstrate the installation, usage, capabilities, and integration of other edge platforms and IoT modules into the EdgeX Foundry platform at a basic level. EdgeX Foundry provides a user-friendly interface that is highly usable for managing and monitoring IoT devices and edge platforms, which are becoming increasingly popular worldwide. For this platform (EdgeX), we will use a Linux (VM) environment. We will then install EdgeX Foundry on our Linux environment using commands from the terminal screen and try to perform operations on the device through specific ports. For these operations to be performed, the ports must be active and if the installation is not done correctly, we will not have access to these ports, so we will not be able to reach the desired interfaces.

EdgeX Foundry has developed itself in recent years and has gone through many updates via the "docker-compose.yml" file. Although the latest version released so far is "hanoi", I have preferred to use the "geneva" version. If you wish, you can perform the operation on other versions, but in this thesis, we will work on the geneva version. The latest current version will be released in the spring of 2021. EdgeX Foundry, which can run on other operating systems, I think runs more stable in the virtual Linux environment that I installed on my computer using the "VirtualBox" tool. After installing EdgeX Foundry, I used the "sudo docker run hello-world" command from the terminal screen to test it and received the output "Hello from Docker!", and from this I saw that the Docker containers and images were loaded correctly.

RESEARCH FINDINGS

In this study where we will send temperature and humidity values, we need to first define them through the "valuedescriptor" section via the "48080" port in Postman. After the value definition, we load our ".yaml" file (the file where the device profile information is located) through the same port (48081). Then, we also send the device information through the same port (48081). After these procedures we perform in Postman, our device information will be displayed on the golang interface of EdgeX Foundry and we can start sending values to our device on EdgeX's golang interface.

CONCLUSIONS AND RECOMMENDATIONS

In this project, the goal of EdgeX Foundry installation, usage, capabilities, and integration of other modules into this platform has been completed. This platform, which is not well-known in Turkey, I am sure that it will be more widely known and used as IoT devices become more widespread. The advancement of technology and the development of smart devices and the need for real-time data transfer will significantly increase the widespread use of this platform. In general, I have tried to explain and introduce this platform with countless functions as best I can. At least by showing the basic concepts and usage methods in this platform, my observations showed me the importance of edge computing. In a project I followed, I became unable to think of a better method for transferring live camera images in real-time. The reason why it is not widely used in Turkey is due to the lack of sufficient interest in IoT devices. Wherever IoT devices are widely used, using EdgeX Foundry will be inevitable. In addition to the need for real-time and continuous data transfer, the need for monitoring these data will increase day by day and EdgeX Foundry will become much more widespread. If you want to learn more about EdgeX Foundry, I suggest you research some Chinese resources. China, one of the countries with the most IoT devices in the world, makes up a significant portion of the resources I researched. I recommend always being in contact with a few people who have information about this platform, which may have resource problems for use, especially. Because, as I mentioned earlier, EdgeX Foundry will eventually become widespread around the world.

RESOURCES

docs.docker.com/engine/install/ubuntu/

docs.docker.com/compose/install/

jonamiki.com/2019/10/11/download-install-and-run-edgex-foundry-in-5-min-on-ubuntu-18-04-server/

Samsung Chief Engineer Suresh LC'nin "Home Edge" Conference

Github Repository (docker and docker compose files)

Youtube "EdgeX Foundry" Channel

Youtube "Gavin Lu" Channel

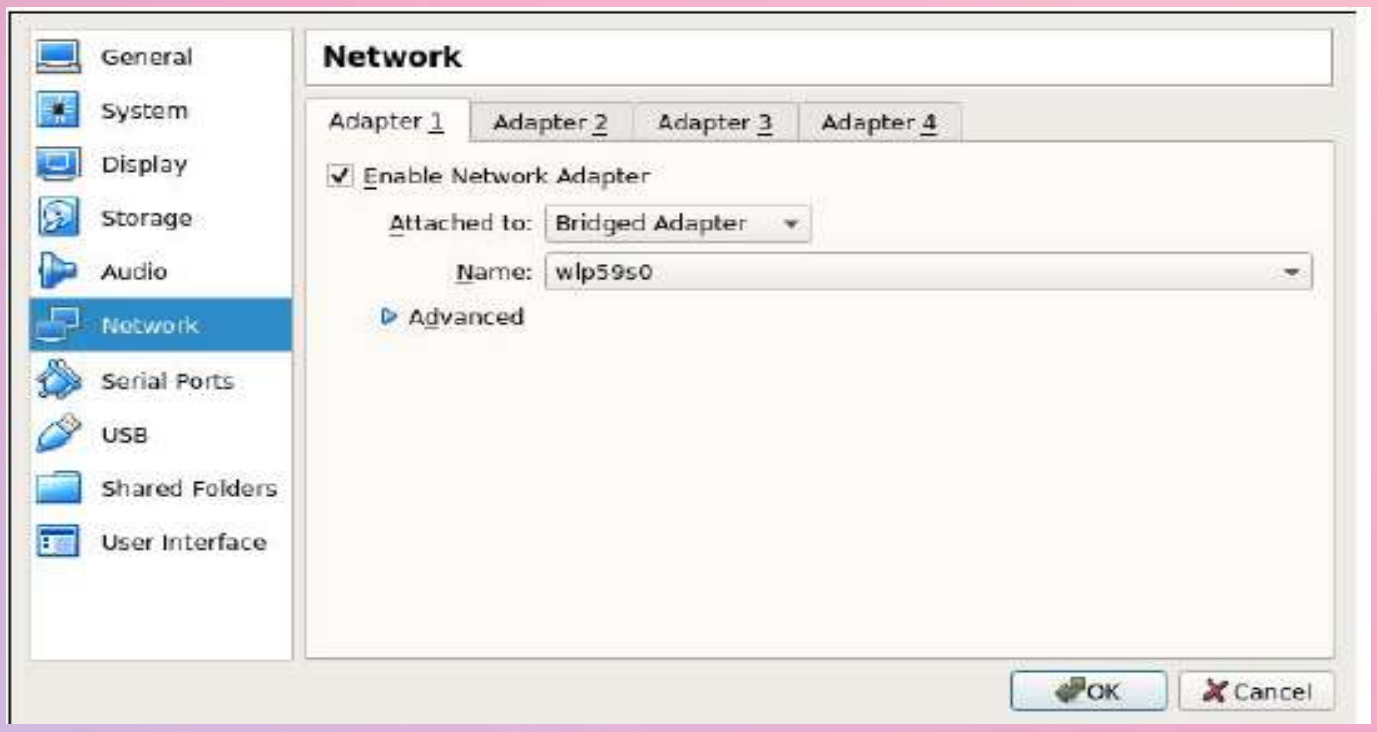
Subrat Sahoo, "EDGEX FOUNDRY AND IT'S SETUP" Article

Subrat Sahoo "SENDING SENSOR DATA TO SERVER USİNG EDGEX FOUNDRY" Article and the .yaml file included in this article

INSTALLING THE EDGEX FOUNDRY

What Is Edgex Foundry?

EdgeX is a platform that allows data collection and control of various IoT and M2M (Many to Many) devices. It has a rules engine that can automate actions while providing real-time access to sensor data. Formatting, compression, and encryption of data to be transmitted for storage and analysis is also possible (e.g. Azure). It is important to note that EdgeX was not designed for production. EdgeX is a platform for managing and monitoring. If EdgeX is running as a Linux VM and you plan to use a microcontroller for sensors (e.g. Raspberry Pi), you need to make a bridge over the network by giving an IP on the "Network" section in the "Settings" tab of the VM. If you are conducting tests through virtual devices, this process is not necessary.



Please note: In all the operations you will perform, you are always expected to enter your Ubuntu password. When entering your Ubuntu password in the terminal screen, no input will be displayed, but you can continue with your operations once you have entered the password correctly and pressed the "Enter" key. Some of the outputs of the screens I show you may not be able to be obtained in full, so I kindly ask you to wait patiently. In some installations, when approval is awaited, you need to press the "e" or "E" key. Please press this key so that you can continue. In Ubuntu operating system, you can paste the texts (such as the commands in this thesis) that you copied (by Ctrl+C) outside the terminal window (in the Windows environment) into the Ubuntu terminal window using the Ctrl+Shift+V (the Ctrl and Shift keys on the left) key combination. Ctrl+V in Windows does not work in the Ubuntu terminal window. Before installation, we remove old versions of Docker (these are: docker, docker.io, docker-engine) if they are installed. Most likely you will not have old versions installed and we will receive answers in the form of "not installed, therefore can not be removed". If one of the old versions is installed, it will be removed with this command. To do this, we will enter the following command in the terminal screen:

sudo apt-get remove docker docker-engine docker.io containerd runc

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo apt-get remove docker docker-engine
 docker.io containerd runc
[sudo] password for edgexfoundry:
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
'docker-engine' kurulu değildi, dolayısıyla kaldırılmadı
'docker' kurulu değildi, dolayısıyla kaldırılmadı
'containerd' kurulu değildi, dolayısıyla kaldırılmadı
'docker.io' kurulu değildi, dolayısıyla kaldırılmadı
'runc' kurulu değildi, dolayısıyla kaldırılmadı
0 paket yükseltilecek, 0 yeni paket kurulacak, 0 paket kaldırılacak ve 0 paket y
ükseltilmeyecek.
```

Next, we need to enter the following commands to update our system.

sudo apt update

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo apt update
[sudo] password for edgexfoundry:
Aynı:  1 http://tr.archive.ubuntu.com/ubuntu bionic InRelease
Aynı:  2 http://tr.archive.ubuntu.com/ubuntu bionic-updates InRelease
Aynı:  3 http://tr.archive.ubuntu.com/ubuntu bionic-backports InRelease
İndir: 4 http://dl.google.com/linux/chrome/deb stable InRelease [1.811 B]
Aynı:  5 http://security.ubuntu.com/ubuntu bionic-security InRelease
İndir: 6 http://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1.090
 B]
1 sn.'de 2.901 B alındı (3.476 B/s)
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
188 paket yükseltilebilir. Bu paketleri görmek için 'apt list --upgradable' komu
tunu çalıştırın.
edgexfoundry@edgexfoundry-VirtualBox:~$
```

sudo apt upgrade -y

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo apt upgrade -y
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Yükseltme hesaplanıyor... Bitti
Aşağıdaki paketler yükseltilecek:
  accountsservice apport apport-gtk apt apt-utils aptdaemon aptdaemon-data
  base-files bind9-host bsdutils busybox-initramfs busybox-static
  ca-certificates dirmngr distro-info-data dnsutils fdisk firefox gdb
  gdbserver gdm3 ghostscript ghostscript-x gir1.2-accountsservice-1.0
  gir1.2-gdm-1.0 gir1.2-javascriptcoregtk-4.0 gir1.2-packagekitglib-1.0
  gir1.2-webkit2-4.0 gnupg gnupg-l10n gnupg-utils gpg gpg-agent gpg-wks-client
  gpg-wks-server gpgconf gpgsm gpgv gstreamer1.0-packagekit imagemagick
  imagemagick-6-common imagemagick-6.q16 initramfs-tools initramfs-tools-bin
  initramfs-tools-core intel-microcode krb5-locales libaccountsservice0
  libapt-inst2.0 libapt-pkg5.0 libbind9-160 libblkid1 libbrotli1 libc-bin
  libc6 libc6-dbg libcryptsetup12 libcurl3-gnutls libdns-export1100 libdns1100
  libevdev2 libexif12 libfdisk1 libfreerdp-client2-2 libfreerdp2-2
  libfreetype6 libgdm1 libgs9 libgs9-common libgssapi-krb5-2 libinput-bin
  libinput10 libirs160 libisc-export169 libisc169 libisccc160 libisccfg160
  libjavascriptcoregtk-4.0-18 libk5crypto3 libkrb5-3 libkrb5support0
  libldap-2.4-2 libldap-common liblwres160 libmagickcore-6.q16-3
  libmagickcore-6.q16-3-extra libmagickwand-6.q16-3 libmount1
  libnss-myhostname libnss-systemd libnss3 libpackagekit-glib2-18
```

Now we install Docker-CE by verifying the certificate.

sudo apt install apt-transport-https ca-certificates curl software-properties-common -y

```
edgexfoundry@edgexfoundry-VirtualBox:~$  sudo apt install apt-transport-https ca
-certificates curl software-properties-common -y
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
ca-certificates zaten en yeni sürümde (20201027ubuntu0.18.04.1).
ca-certificates elle kurulmuş olarak ayarlandı.
software-properties-common zaten en yeni sürümde (0.96.24.32.14).
software-properties-common elle kurulmuş olarak ayarlandı.
Aşağıdaki ek paketler kurulacak:
  libcurl4
Aşağıdaki YENİ paketler kurulacak:
  apt-transport-https curl libcurl4
0 paket yükseltilecek, 3 yeni paket kurulacak, 0 paket kaldırılacak ve 0 paket y
ükseltilmeyecek.
375 kB arşiv dosyası indirilecek.
Bu işlem tamamlandıktan sonra 1.191 kB ek disk alanı kullanılacak.
İndir: 1 http://tr.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 apt-t
ransport-https all 1.6.12ubuntu0.2 [1.696 B]
İndir: 2 http://tr.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcurl4
amd64 7.58.0-2ubuntu3.12 [214 kB]
İndir: 3 http://tr.archive.ubuntu.com/ubuntu bionic-updates/main amd64 curl amd6
4 7.58.0-2ubuntu3.12 [159 kB]
```

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

```
edgexfoundry@edgexfoundry-VirtualBox:~$ curl -fsSL https://download.docker.com/l
inux/ubuntu/gpg | sudo apt-key add -
OK
```

The output of this process is "OK".

Next, we install the Docker repository based on our Linux version. To find out our Linux version, we use the following command.

lsb_release -a

```
edgexfoundry@edgexfoundry-VirtualBox:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.5 LTS
Release:        18.04
Codename:       bionic
```

As you can see, my Linux version name is "bionic".

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

In this command, we write the part that says "bionic" next to "Codename" in the output screen I gave above. For example, some Linux distributions are "focal" while mine is "bionic". If your Linux distribution is "focal", the command you need to enter into the terminal screen will look like this (only the "bionic" part will change).

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo add-apt-repository "deb [arch=amd64
] https://download.docker.com/linux/ubuntu bionic stable"
Aynı:  1 http://dl.google.com/linux/chrome/deb stable InRelease
Aynı:  2 http://security.ubuntu.com/ubuntu bionic-security InRelease
Aynı:  3 http://tr.archive.ubuntu.com/ubuntu bionic InRelease
Aynı:  4 http://tr.archive.ubuntu.com/ubuntu bionic-updates InRelease
Aynı:  5 http://tr.archive.ubuntu.com/ubuntu bionic-backports InRelease
İndir: 6 https://download.docker.com/linux/ubuntu bionic InRelease [64,4 kB]
İndir: 7 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [
14,9 kB]
1 sn.'de 79,3 kB alındı (59,6 kB/s)
Paket listeleri okunuyor... Bitti
```

Next, we perform a system update.

sudo apt update

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo apt update
Aynı:  1 http://tr.archive.ubuntu.com/ubuntu bionic InRelease
Aynı:  2 http://tr.archive.ubuntu.com/ubuntu bionic-updates InRelease
Aynı:  3 http://tr.archive.ubuntu.com/ubuntu bionic-backports InRelease
Aynı:  4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Aynı:  5 http://dl.google.com/linux/chrome/deb stable InRelease
Aynı:  6 https://download.docker.com/linux/ubuntu bionic InRelease
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Tüm paketler güncel.
```

We will now install Docker-CE.

sudo apt install docker-ce -y

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo apt install docker-ce -y
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Aşağıdaki ek paketler kurulacak:
  containerd.io docker-ce-cli docker-ce-rootless-extras git git-man
  liberror-perl pigz
Önerilen paketler:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit
  git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
Tavsiye edilen paketler:
  slirp4netns
Aşağıdaki YENİ paketler kurulacak:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras git git-man
  liberror-perl pigz
0 paket yükseltilecek, 8 yeni paket kurulacak, 0 paket kaldırılacak ve 0 paket y
ükseltilmeyecek.
108 MB arşiv dosyası indirilecek.
Bu işlem tamamlandıktan sonra 484 MB ek disk alanı kullanılacak.
İndir: 1 http://tr.archive.ubuntu.com/ubuntu bionic/universe amd64 pigz amd64 2.
4-1 [57,4 kB]
İndir: 2 http://tr.archive.ubuntu.com/ubuntu bionic/main amd64 liberror-perl all
 0.17025-1 [22,8 kB]
```

Now, we grant the necessary execution permissions to the current user (the output of this command is not present) and we restart the linux environment to activate these permissions.

sudo usermod -aG docker ${USER}

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo usermod -aG docker ${USER}
```

We are now installing Docker Compose.

sudo apt install docker-compose -y

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo apt install docker-compose -y
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Aşağıdaki ek paketler kurulacak:
  cgroupfs-mount golang-docker-credential-helpers libpython-stdlib python
  python-asn1crypto python-backports.ssl-match-hostname python-cached-property
  python-certifi python-cffi-backend python-chardet python-cryptography
  python-docker python-dockerpty python-dockerpycreds python-docopt
  python-enum34 python-funcsigs python-functools32 python-idna
  python-ipaddress python-jsonschema python-minimal python-mock python-openssl
  python-pbr python-pkg-resources python-requests python-six python-texttable
  python-urllib3 python-websocket python-yaml python2.7 python2.7-minimal
Önerilen paketler:
  python-doc python-tk python-cryptography-doc python-cryptography-vectors
  python-enum34-doc python-funcsigs-doc python-mock-doc python-openssl-doc
  python-openssl-dbg python-setuptools python-socks python-ntlm python2.7-doc
  binfmt-support
Tavsiye edilen paketler:
  docker.io
```

After installing Docker Compose, we create a directory to work in, and then enter that directory (the output of this command is not available).

mkdir edgex

cd edgex

```
edgexfoundry@edgexfoundry-VirtualBox:~$ mkdir edgex
edgexfoundry@edgexfoundry-VirtualBox:~$ cd edgex
```

After completing this process, we will continue to work in the "edgex" directory. The terminal screen will look like this, and from now on we will perform our docker operations in this screen:

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$
```

We will then download the "geneva" version of the docker compose file. If the following command appears as 2 lines, make sure you have written it side by side in the linux terminal screen.

wget https://raw.githubusercontent.com/edgexfoundry/developer-scripts/master/releases/geneva/compose-files/docker-compose-geneva-redis.yml

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ wget https://raw.githubusercontent
.com/edgexfoundry/developer-scripts/master/releases/geneva/compose-files/docker-
compose-geneva-redis.yml
--2021-01-04 23:48:13--  https://raw.githubusercontent.com/edgexfoundry/develope
r-scripts/master/releases/geneva/compose-files/docker-compose-geneva-redis.yml
raw.githubusercontent.com (raw.githubusercontent.com) çözümleniyor... 151.101.11
2.133
raw.githubusercontent.com (raw.githubusercontent.com)[151.101.112.133]:443 bağla
nılıyor... bağlantı kuruldu.
HTTP isteği gönderildi, cevap bekleniyor... 200 OK
Uzunluk: 18551 (18K) [text/plain]
Kayıt yeri: `docker-compose-geneva-redis.yml'

          docker-co   0%[                    ]       0  --.-KB/s               P
arçalama arızası (çekirdek döküldü)
```

The "breakage error (kernel dumped)" error that we received in this command does not pose a problem for us. Even if we received this error, our docker-compose file will have been successfully downloaded. The "200 OK" output is a sign that we made a correct call.

After this process, when we enter the "ls" command, we will be able to view the "docker-compose-geneva-redis.yml" file.

ls

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ ls
docker-compose-geneva-redis.yml
```

Next, we copy this file with the name "docker-compose.yml".

cp docker-compose-geneva-redis.yml docker-compose.yml

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ cp docker-compose-geneva-redis.yml
 docker-compose.yml
```

We can pull the docker containers by using the following command:

docker-compose pull

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ docker-compose pull
Pulling security-secrets-setup (edgexfoundry/docker-edgex-secrets-setup-go:1.2.1
)...
ERROR: Couldn't connect to Docker daemon at http+docker://localunixsocket - is i
t running?

If it's at a non-standard location, specify the URL with the DOCKER_HOST environ
ment variable.
```

If you encounter such an error, restart your Linux environment. Go to the directory where Docker is installed in your terminal screen (in this example, the edgex directory) by entering the "cd edgex" command and enter the same command again. If you did not receive an error, you can continue with the operations.

```
edgexfoundry@edgexfoundry-VirtualBox:~$ cd edgex/
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ docker-compose pull
Pulling security-secrets-setup (edgexfoundry/docker-edgex-secrets-setup-go:1.2.1
)...
1.2.1: Pulling from edgexfoundry/docker-edgex-secrets-setup-go
df20fa9351a1: Downloading [>                                                  ]
 29.17kB/2.798MBlling fs layer
d73780d9ac1e: Pulling fs layer
df20fa9351a1: Downloading [=>                                                 ]
 59.25kB/2.798MBwnload complete
df20fa9351a1: Downloading [==>                                                ]
   147kB/2.798MBiting
df20fa9351a1: Pull complete
575d4ce6ad7c: Pull complete
d73780d9ac1e: Pull complete
d6122b2a9a2e: Pull complete
0fa9e698c317: Pull complete
8cac0e3883cb: Pull complete
e6994af019a8: Pull complete
f4a1c249a45c: Pull complete
647965e227e7: Pull complete
308bad0a7dc2: Pull complete
d592bcfae5a7: Pull complete
Digest: sha256:254ed80e8d785d1272201bf77bb7e04cd73a4f5cccea069d3b6231877681932f
```

To start EdgeX using docker, you will run the following command:

docker-compose up -d

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ docker-compose up -d
Creating network "edgex_default" with the default driver
Creating network "edgex_edgex-network" with driver "bridge"
Creating volume "edgex_db-data" with default driver
Creating volume "edgex_vault-logs" with default driver
Creating volume "edgex_consul-data" with default driver
Creating volume "edgex_vault-init" with default driver
Creating volume "edgex_consul-config" with default driver
Creating volume "edgex_secrets-setup-cache" with default driver
Creating volume "edgex_consul-scripts" with default driver
Creating volume "edgex_log-data" with default driver
Creating volume "edgex_vault-file" with default driver
Creating volume "edgex_vault-config" with default driver
Creating edgex-secrets-setup ...
Creating edgex-secrets-setup ... done
Creating kong-db ...
Creating edgex-core-consul ...
Creating edgex-core-consul
Creating edgex-core-consul ... done
Creating kong-migrations ...
Creating edgex-vault ...
Creating edgex-vault
Creating kong-migrations ... done
Creating kong ...
Creating edgex-vault ... done
Creating edgex-vault-worker ...
Creating edgex-vault-worker ... done
Creating edgex-redis ...
Creating kong ... done
Creating edgex-proxy ...
```

The following command can be used to list the running containers:

docker-compose ps

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ docker-compose ps
         Name                    Command           State            Ports
-------------------------------------------------------------------------------
edgex-app-service-       /app-service-             Up       48095/tcp, 127.0.0.1:
configurable-rules       configurable  ...                   48100->48100/tcp
edgex-core-command       /core-command             Up       127.0.0.1:48082->4808
                         -cp=consul.h ...                   2/tcp
edgex-core-consul        edgex-consul-             Up       8300/tcp, 8301/tcp,
                         entrypoint.sh ...                  8301/udp, 8302/tcp,
                                                            8302/udp, 127.0.0.1:8
                                                            400->8400/tcp, 127.0.
                                                            0.1:8500->8500/tcp,
                                                            8600/tcp, 8600/udp
edgex-core-data          /core-data                Up       127.0.0.1:48080->4808
                         -cp=consul.http ...                0/tcp, 127.0.0.1:5563
                                                            ->5563/tcp
edgex-core-metadata      /core-metadata            Up       127.0.0.1:48081->4808
                         -cp=consul. ...                    1/tcp
edgex-device-rest        /device-rest-go           Up       127.0.0.1:49986->4998
                         --cp=consu ...                     6/tcp
edgex-device-virtual     /device-virtual           Up       127.0.0.1:49990->4999
                         --cp=consu ...                     0/tcp
edgex-kuiper             /usr/bin/docker-          Up       127.0.0.1:20498->2049
                         entrypoint ...                     8/tcp, 127.0.0.1:4807
                                                            5->48075/tcp,
                                                            9081/tcp
```

You can interact with EdgeX through the terminal screen or browser. You can list the registered devices using the command below.

curl http://localhost:48082/api/v1/device

We can use "jq" to format the output of the command to make it more readable. To do so, we first need to install "jq".

sudo apt install jq

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ sudo apt install jq
[sudo] password for edgexfoundry:
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Aşağıdaki ek paketler kurulacak:
  libjq1 libonig4
Aşağıdaki YENİ paketler kurulacak:
  jq libjq1 libonig4
0 paket yükseltilecek, 3 yeni paket kurulacak, 0 paket kaldırılacak ve 0 paket yükseltilmeyece
k.
276 kB arşiv dosyası indirilecek.
Bu işlem tamamlandıktan sonra 930 kB ek disk alanı kullanılacak.
N: '/etc/apt/sources.list.d/' dizinindeki 'google-chrome.list.save' dosyası geçersiz bir dosya
 uzantısı olduğu için yok sayılıyor
Devam etmek istiyor musunuz? [E/h] e
İndir: 1 http://tr.archive.ubuntu.com/ubuntu bionic/universe amd64 libonig4 amd64 6.7.0-1 [119
 kB]
İndir: 2 http://tr.archive.ubuntu.com/ubuntu bionic/universe amd64 libjq1 amd64 1.5+dfsg-2 [11
1 kB]
İndir: 3 http://tr.archive.ubuntu.com/ubuntu bionic/universe amd64 jq amd64 1.5+dfsg-2 [45,6 k
B]
1 sn.'de 276 kB alındı (246 kB/s)
N: '/etc/apt/sources.list.d/' dizinindeki 'google-chrome.list.save' dosyası geçersiz bir dosya
 uzantısı olduğu için yok sayılıyor
Daha önce seçili olmayan libonig4:amd64 paketi seçiliyor.
(Veritabanı okunuyor ... 166871 dosya veya dizin kurulu durumda.)
Paket açılacak: .../libonig4_6.7.0-1_amd64.deb ...
Paket açılıyor: libonig4:amd64 (6.7.0-1) ...
Daha önce seçili olmayan libjq1:amd64 paketi seçiliyor.
Paket açılacak: .../libjq1_1.5+dfsg-2_amd64.deb
```

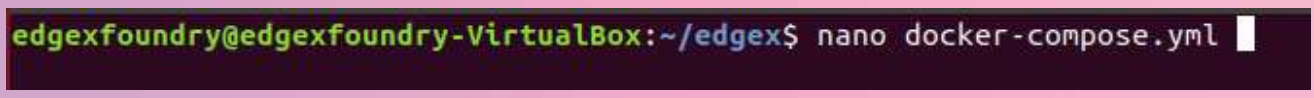
You can use the following command to get the formatted output.

curl http://localhost:48082/api/v1/device | jq

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ curl http://localhost:48082/api/v1/device | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  9873    0  9873    0     0   132k      0 --:--:-- --:--:-- --:--:--  132k
[
  {
    "id": "1524f882-f891-431b-9f49-1b69006f2bd4",
    "name": "Random-Binary-Device",
    "adminState": "UNLOCKED",
    "operatingState": "ENABLED",
    "labels": [
      "device-virtual-example"
    ],
    "commands": [
      {
        "created": 1609794678870,
        "modified": 1609794678870,
        "id": "1e0e45cb-867e-4d07-a787-8dd9a8a768e1",
        "name": "Binary",
        "get": {
          "path": "/api/v1/device/{deviceId}/Binary",
          "responses": [
            {
              "code": "200",
```

After viewing the registered devices, we may want to access an EdgeX interface. There are many user interfaces for EdgeX, one of which is the "Golang UI", which is EdgeX's own interface. To access this interface, we need to do the following:

We open the "docker-compose.yml" file located in the "edgex" directory in a text editor (e.g., nano) in this example.



We add an entry for the Golang user interface just below the "services" section by scrolling down. This entry will look like this:

```
ui:
  container_name: edgex-ui-go
  hostname: edgex-ui-go
  image:
    nexus3.edgexfoundry.org:10004/docker-edgex-ui-go:master
  networks:
    edgex-network: null
  ports:
  - "0.0.0.0:4000:4000/tcp"
  read_only: true
```

It is important to pay attention to the spaces here.

2 spaces before "ui:".
4 spaces before "container_name: edgex-ui-go".
4 spaces before "hostname: edgex-ui-go".
4 spaces before "image:".
6 spaces before "nexus3.edgexfoundry.org:10004/docker-edgex-ui-go:master".
4 spaces before "networks:".
6 spaces before "edgex-network: null".
4 spaces before "ports:".
4 spaces before "- "0.0.0.0:4000:4000/tcp".
4 spaces before "read_only: true".

At the end, the screen should look like this:

```
services:
  ui:
    container_name: edgex-ui-go
    hostname: edgex-ui-go
    image:
      nexus3.edgexfoundry.org:10004/docker-edgex-ui-go:master
    networks:
      edgex-network: null
    ports:
    - "0.0.0.0:4000:4000/tcp"
    read_only: true
```

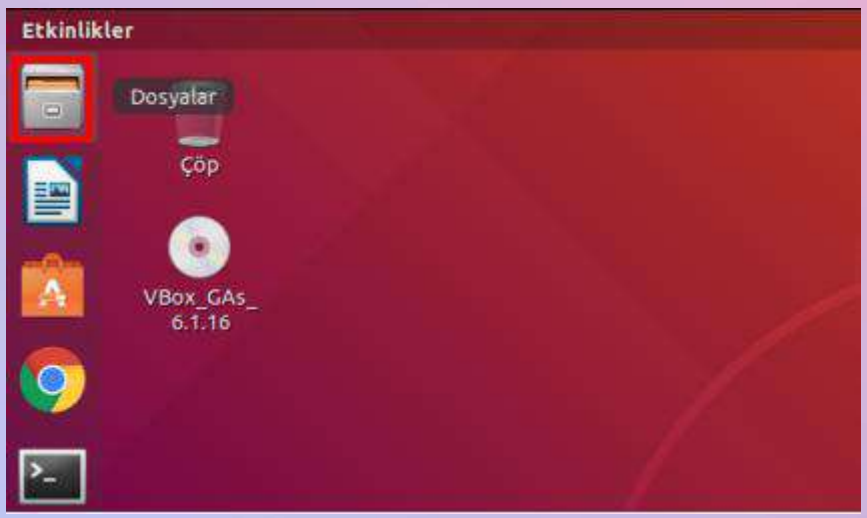
After adding the entry, press "Ctrl" on the left side of the keyboard followed by "X" to exit. A screen like the following will appear:

```
Değiştirilen tampon kaydeidlsin mi?  ("Hayır" demek değişiklikleri SİLECEK.)
 E Evet
 H Hayır         ^C İptal
```

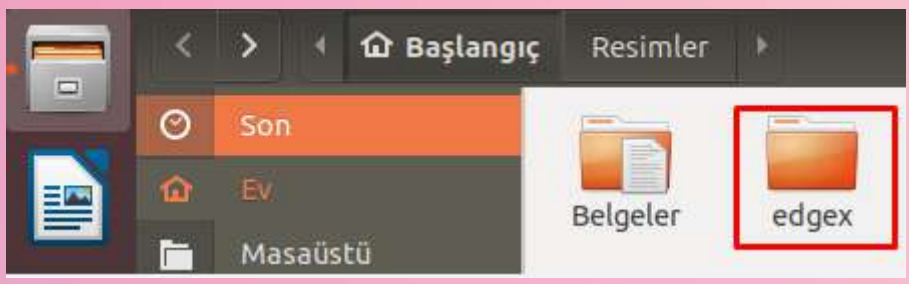
Then we press the "Y" key on the left side of our keyboard, and it will ask us if we want to change the file name. We just press the "Enter" key, and we finish the editing.

```
Yazılacak Dosya Adı: docker-compose.yml
^G Yardım Al          M-D DOS Biçimi      M-A Sonuna Ekle     M-B Yedek Dosyası
^C İptal              M-M Mac Biçimi      M-P Başına Ekle     ^T Dosyalara
```

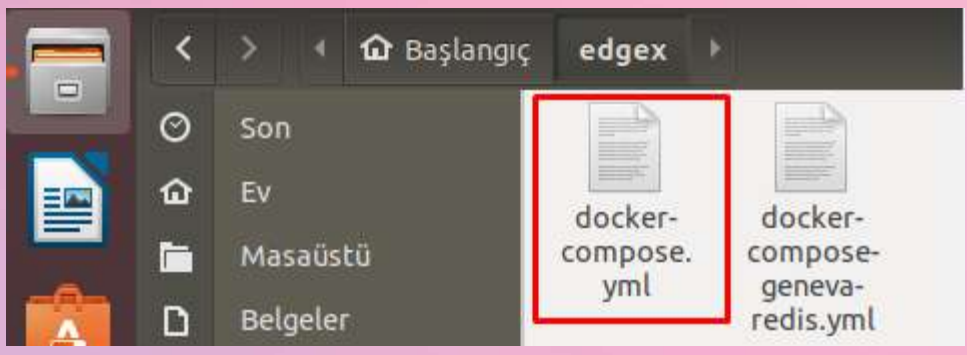
Alternatively, we have the chance to perform this operation easily in the Ubuntu interface. To do this, we first go to the "Files" section in the Ubuntu interface.



Then, we enter the "edgex" directory located there.



We open the "docker-compose.yml" file in the "edgex" directory.



You can easily add the entry for the Golang user interface by going to the "services" section, paying attention to the spaces, and adding the entry with ease.

```
services:
  ui:
    container_name: edgex-ui-go
    hostname: edgex-ui-go
    image:
      nexus3.edgexfoundry.org:10004/docker-edgex-ui-go:master
    networks:
      edgex-network: null
    ports:
    - "0.0.0.0:4000:4000/tcp"
    read_only: true

  consul:
    image: edgexfoundry/docker-edgex-consul:1.2.0
    ports:
      - "127.0.0.1:8400:8400"
      - "127.0.0.1:8500:8500"
    container_name: edgex-core-consul
    hostname: edgex-core-consul
    networks:
      - edgex-network
    volumes:
      - consul-config:/consul/config:z
      - consul-data:/consul/data:z
```

After adding the Golang user interface entry, we enter the following command to load it:

docker-compose up -d

```
edgexfoundry@edgexfoundry-VirtualBox:~/edgex$ docker-compose up -d
Pulling ui (nexus3.edgexfoundry.org:10004/docker-edgex-ui-go:master)...
master: Pulling from docker-edgex-ui-go
801bfaa63ef2: Downloading [>                                                  ]
801bfaa63ef2: Downloading [=>                                                 ]
 64.92kB/2.799MB
801bfaa63ef2: Downloading [=>                                                 ]
801bfaa63ef2: Downloading [==>                                                ]
801bfaa63ef2: Downloading [====>                                              ]
 228.8kB/2.799MB
801bfaa63ef2: Downloading [========>                                          ]
801bfaa63ef2: Downloading [==========>                                        ]
801bfaa63ef2: Downloading [=============>                                     ]
801bfaa63ef2: Downloading [================>                                  ]
 851.4kB/2.799MB
801bfaa63ef2: Downloading [==================>                                ]
 916.9kB/2.799MB
801bfaa63ef2: Downloading [====================>                              ]
801bfaa63ef2: Downloading [======================>                            ]
 1.113MB/2.799MB
801bfaa63ef2: Downloading [========================>                          ]
801bfaa63ef2: Pull complete
25e411ad8891: Extracting [>                                                   ]
 98.3kB/6.965MBB
```

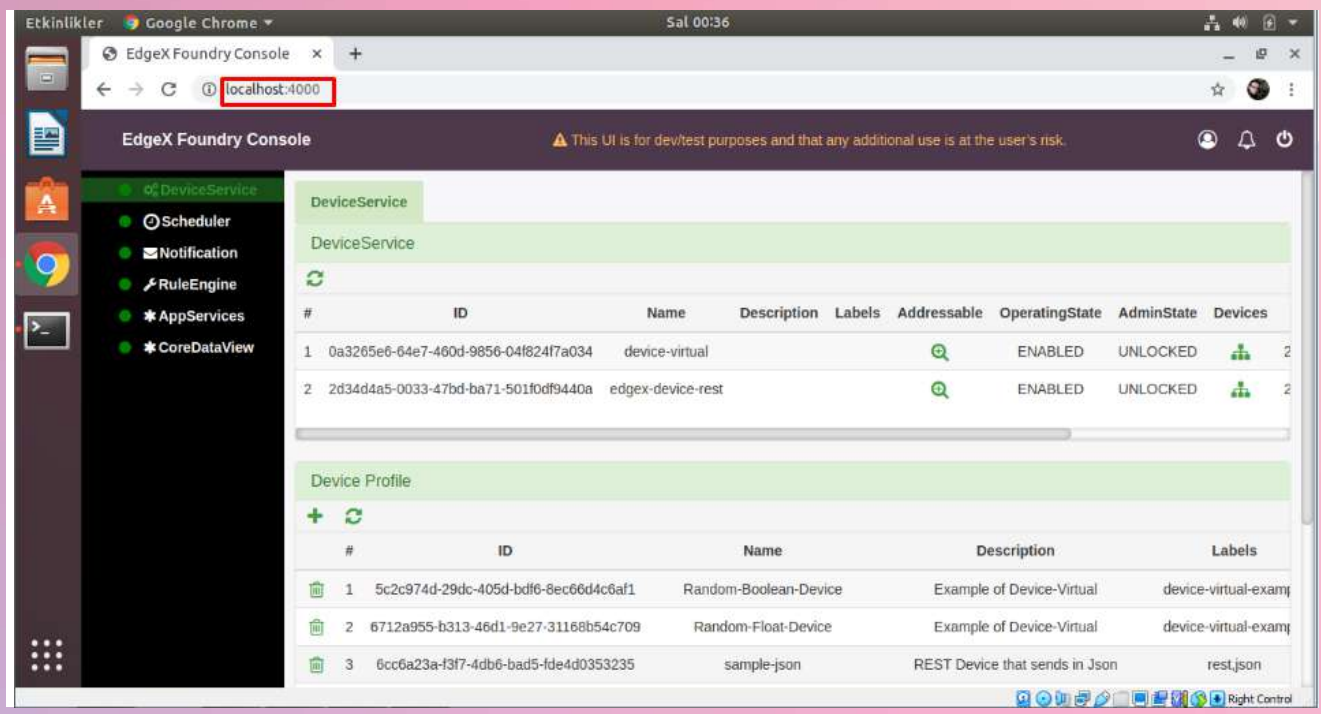
"We are waiting for it to load and once it's completed, we will be able to access the EdgeX Foundry's own interface called "Golang UI" via a browser. To do this, we open our browser (Google Chrome, Firefox, etc.) and in the search bar, type:

localhost:4000

It is necessary to enter. If asked:

Name : admin
Password:admin

You can log in with, but most likely you will have directly reached the Golang interface without being asked from you.


Etkinlikler
Google Chrome
Sal 00:36
EdgeX Foundry Console
localhost:4000
EdgeX Foundry Console
This UI is for dev/test purposes and that any additional use is at the user's risk.
DeviceService
Scheduler
Notification
RuleEngine
AppServices
CoreDataView
DeviceService
DeviceService
ID Name Description Labels Addressable OperatingState AdminState Devices
1 0a3265e6-64e7-460d-9856-04f824f7a034 device-virtual ENABLED UNLOCKED
2 2d34d4a5-0033-47bd-ba71-501f0df9440a edgex-device-rest ENABLED UNLOCKED
Device Profile
ID Name Description Labels
1 5c2c974d-29dc-405d-bdf6-8ec66d4c6af1 Random-Boolean-Device Example of Device-Virtual device-virtual-exam
2 6712a955-b313-46d1-9e27-31168b54c709 Random-Float-Device Example of Device-Virtual device-virtual-exam
3 6cc6a23a-f3f7-4db6-bad5-fde4d0353235 sample-json REST Device that sends in Json rest,json
Right Control

We had mentioned that there are many interfaces for EdgeX. One of them is "Postman". To install Postman, we need to enter the following command in the terminal screen of Ubuntu (through the snap package installer):

sudo snap install postman

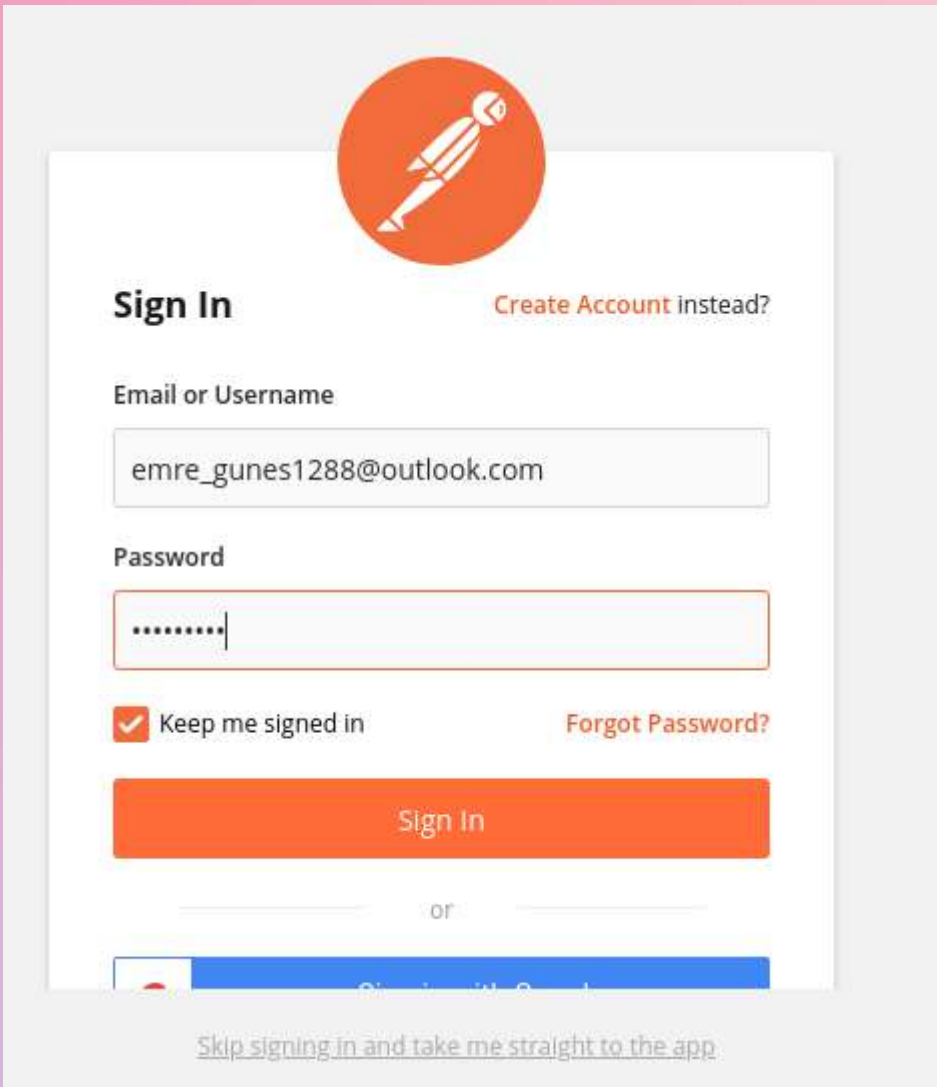
```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo snap install postman
[sudo] password for edgexfoundry:
Download snap "core" (10577) from channel "stable"          65%  986kB/s 36.1s
```

the installation process is successful, you should get an output like this:

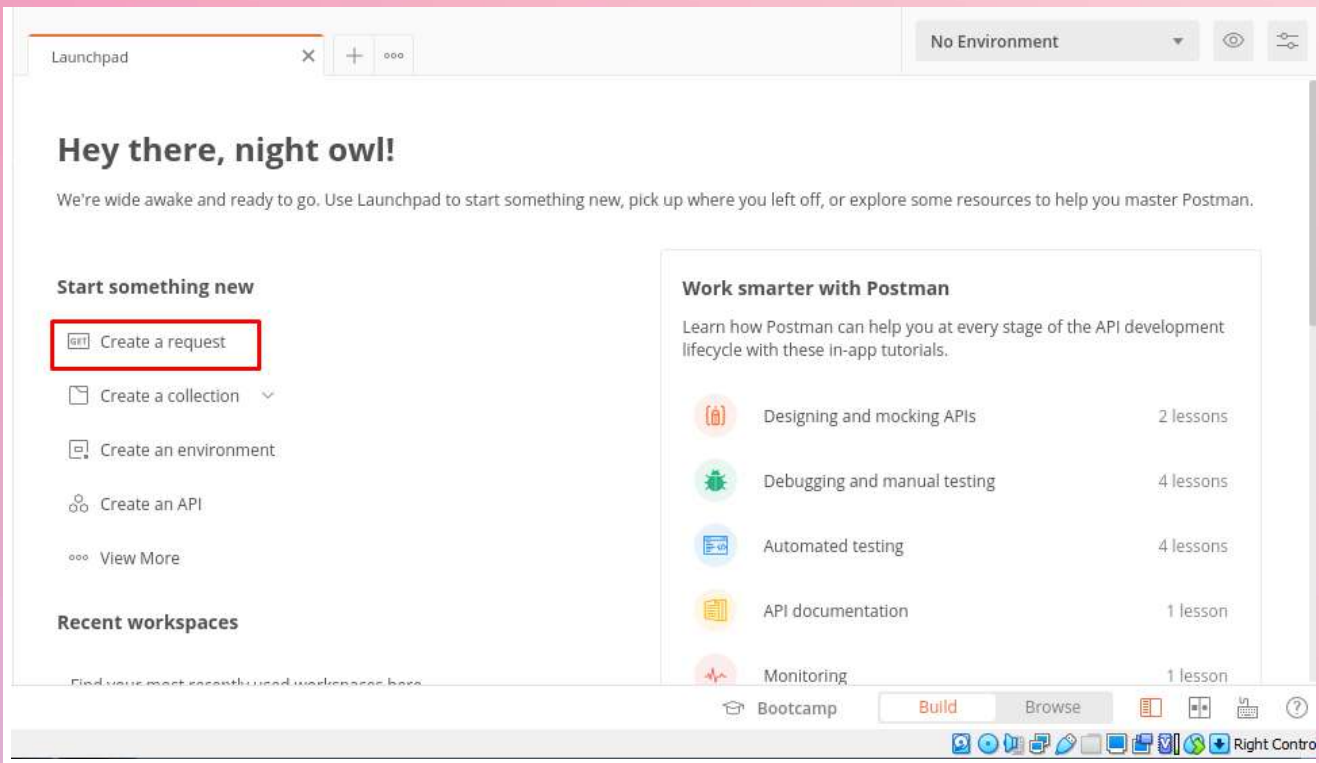
```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo snap install postman
[sudo] password for edgexfoundry:
postman 7.36.1 from Postman, Inc. (postman-inc✓) installed
```

To open Postman in the Ubuntu interface, we first need to click on the "Show Applications" icon. Then, we need to either view all the applications or type "Postman" in the search bar, view the application, and click on it.

After Postman has opened, you need to sign up with a username and password in order to save the operations.

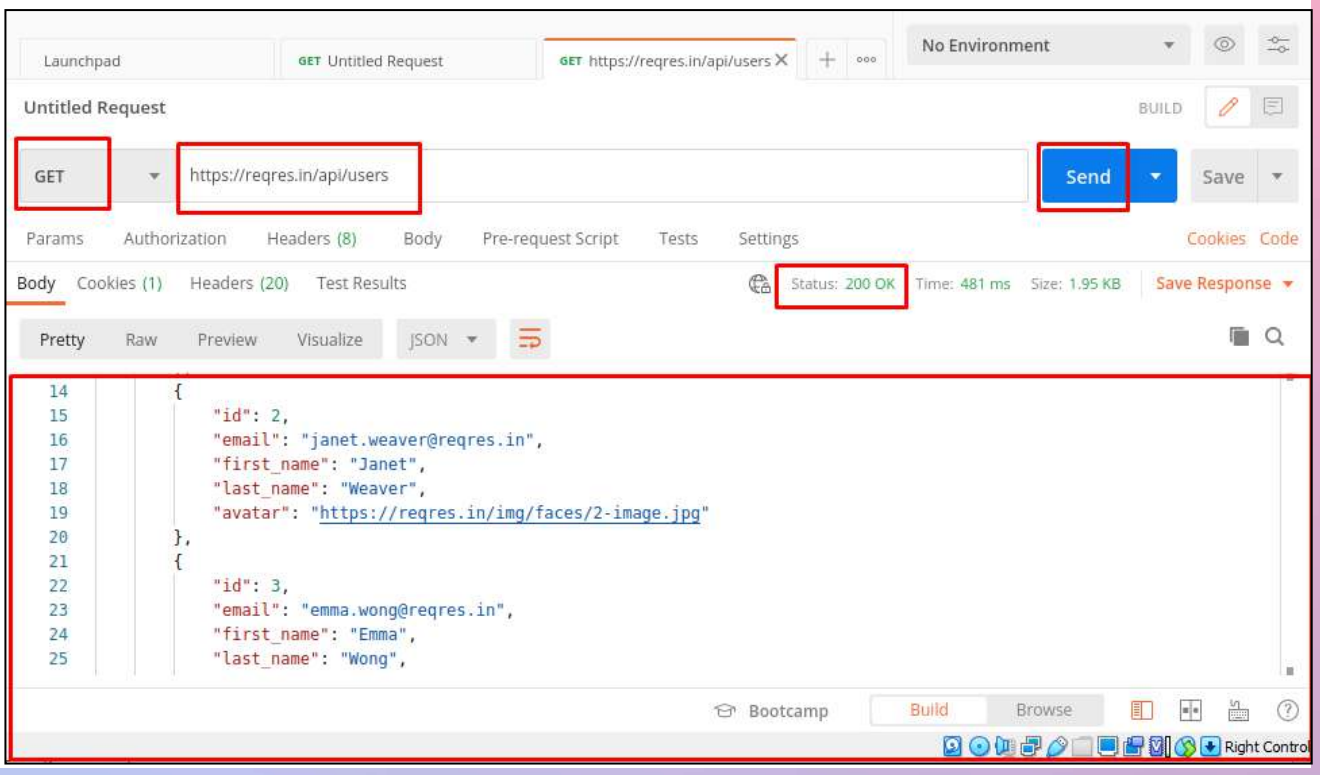


After logging in, we need to click on the "Create a Request" button to perform operations.



To test if EdgeX is working, we need to make a "test request" by using the "GET" method and entering the following address, then clicking on the "Send" button. If the test is successful, a JSON result containing a series of users will be returned.

https://reqres.in/api/users



In Postman, if we are making successful requests, the "Status" section will give a "200 OK" output. If we have made it this far without any issues, it means that we have successfully run the EdgeX Foundry platform.

STOP (REMOVE) EDGEX

If you want to stop EdgeX Foundry, there are several options available. They are:

To stop containers only:

docker-compose stop

To stop and remove containers:

docker-compose down

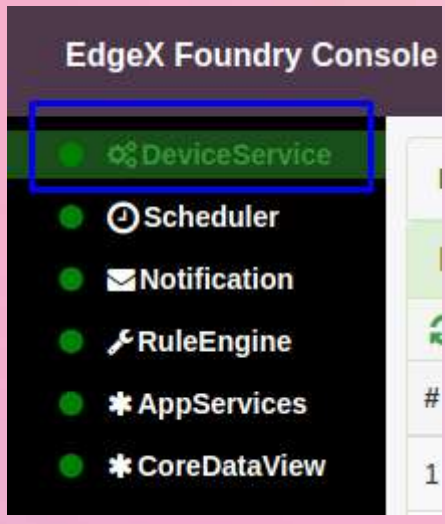
To remove containers and volumes:

docker-compose down -v

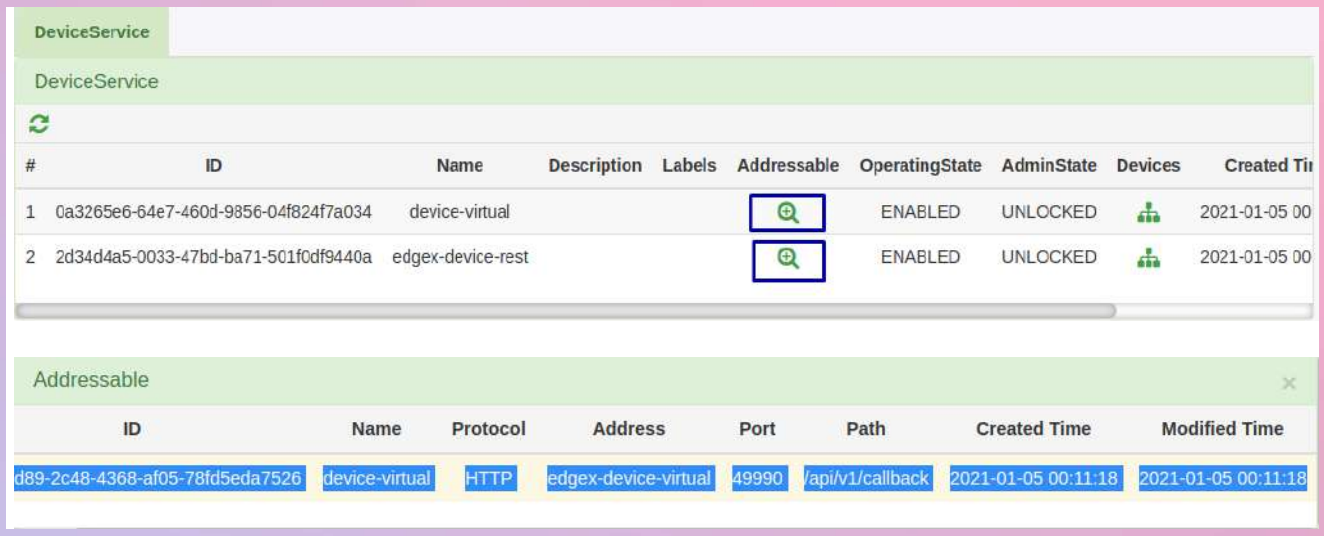
USING THE EDGEX FOUNDRY

After our operations, by entering "localhost:4000" in our web browser, we will access the EdgeX Golang UI, which is EdgeX's own user interface, and we will be able to manage and monitor the ready-made devices.

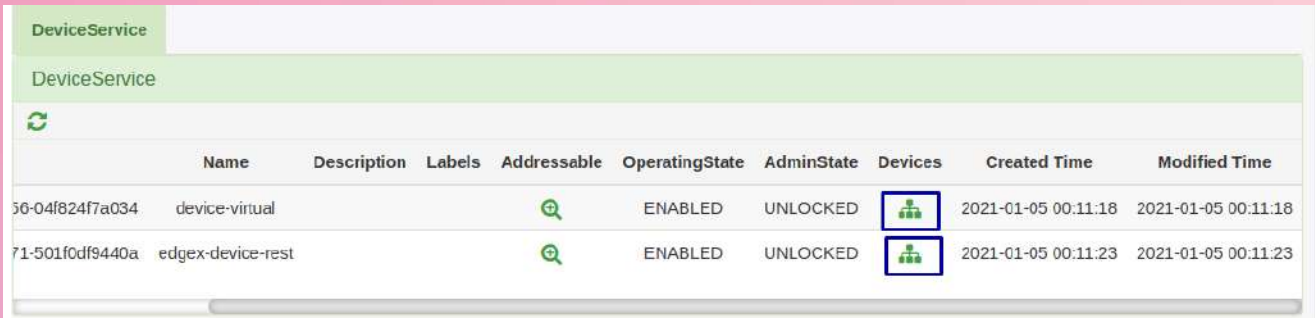
DEVICE SERVICE



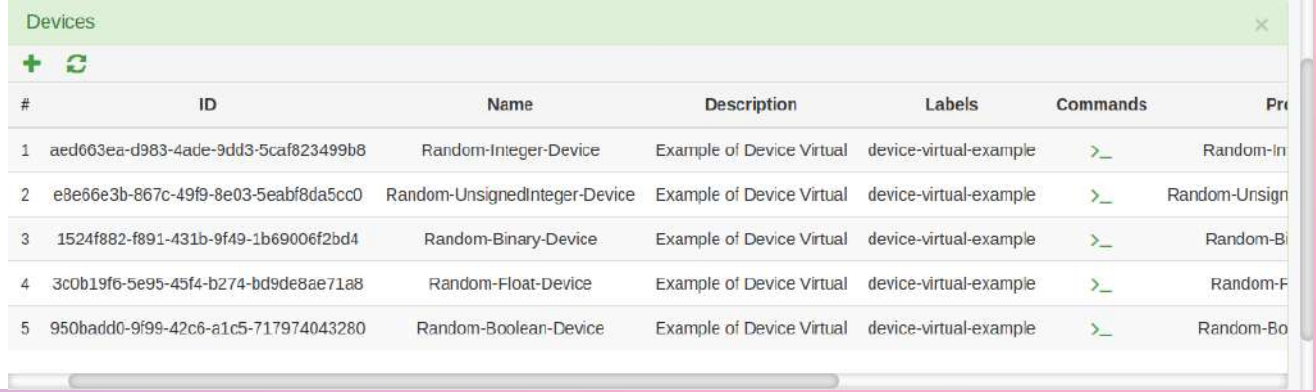
The Device Service is used to manage all device services and devices. The "Addressable" button on the Device Service opens a small window that shows some information for this device service.



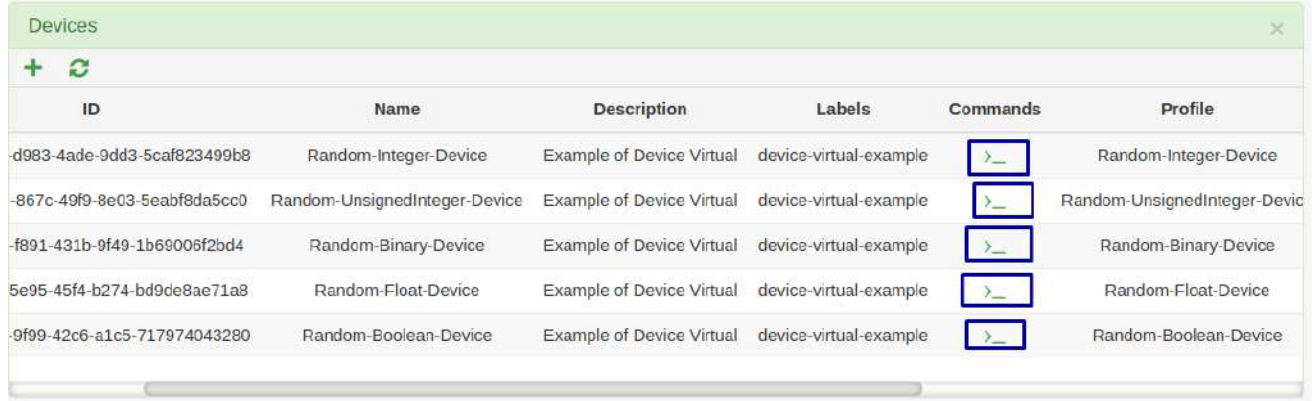
With the "Devices" button on the Device Service, we can list the devices and manage them.



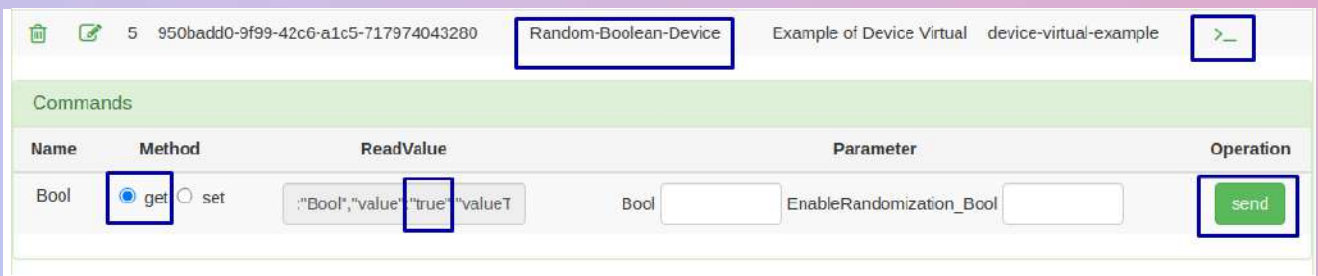
When we click on the Devices button, a new window opens immediately below it and the devices are displayed in this window.



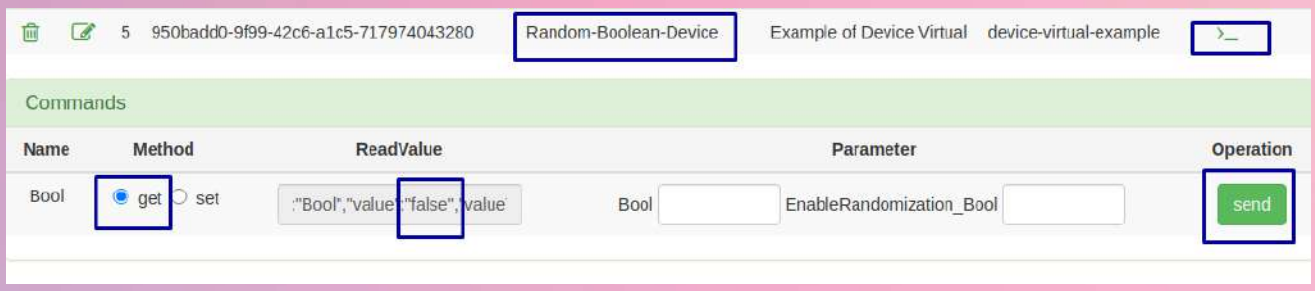
We can send commands to the devices using the "Commands" buttons in the Devices window.



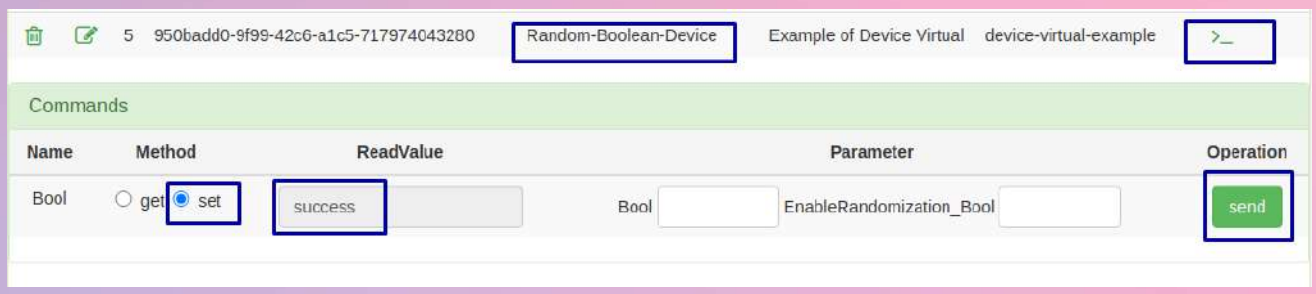
When you click the Commands button, a new window opens immediately below it for the device you want to send a command to.



For example, if we click on the "Commands" button next to the device named "Random-Boolean-Device" above, a new window opens. In this window, we can generate a random boolean value by using the "get" method and pressing the "send" button. In this example, the random value generated is "true". However, since this is a random boolean value, there is also a chance of getting "false". By trying a few times, we can also see "false". As we can see from the screenshot below, depending on our luck, "false" can also be generated.

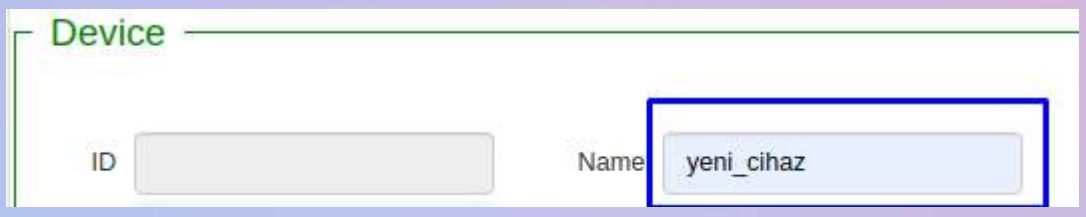


In this screen, we can also send our desired value to the device by using the "set" method. We can send these parameters to the device by entering the necessary parameters in the blank spaces in the "Parameter" section.

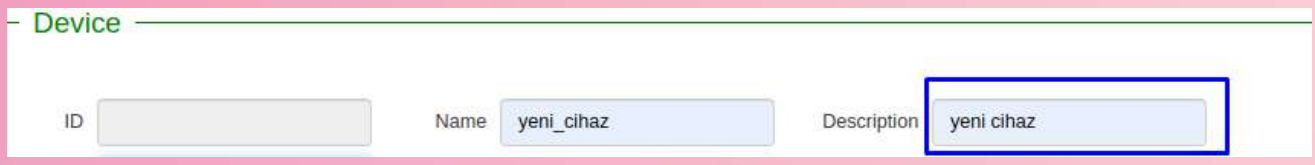


This device generates a value automatically, so there is no need to enter a value, but if you want to send a value, if your request is successfully completed, you will receive a "success" output in the "ReadValue" section. To create a new device based on a specific device profile, you can click the "+" icon in the "Devices" table, create the necessary parameters, and then add the device by clicking the "✔" icon. From the "Device Service" tab, you can select either the "virtual" or "rest" device service.

In the "Device" tab, the "ID" will be automatically generated in a unique manner. In the "Name" section, we can name our device.



In the "Description" section, you can add an optional description.

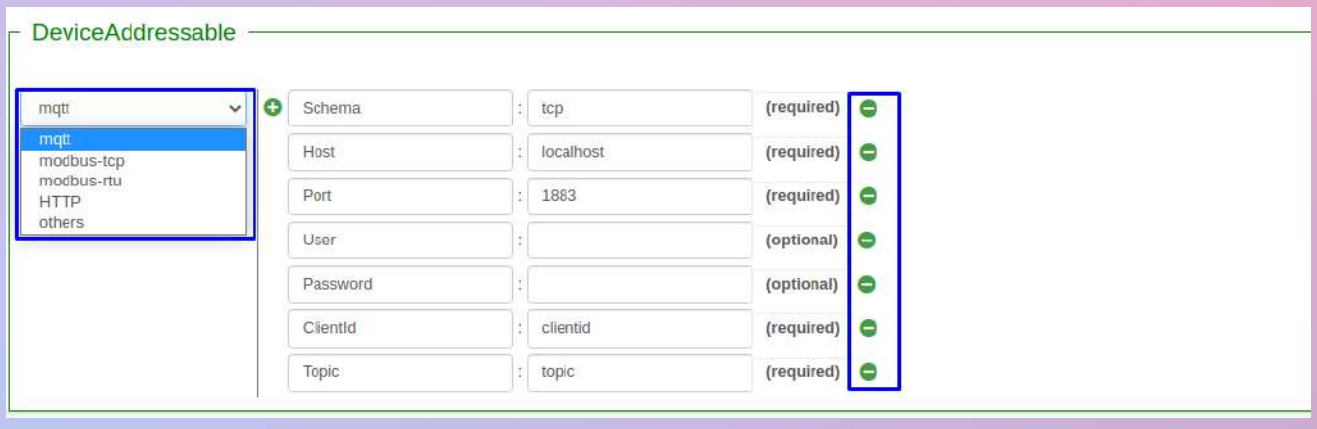


We can label our device in the "Label" section.

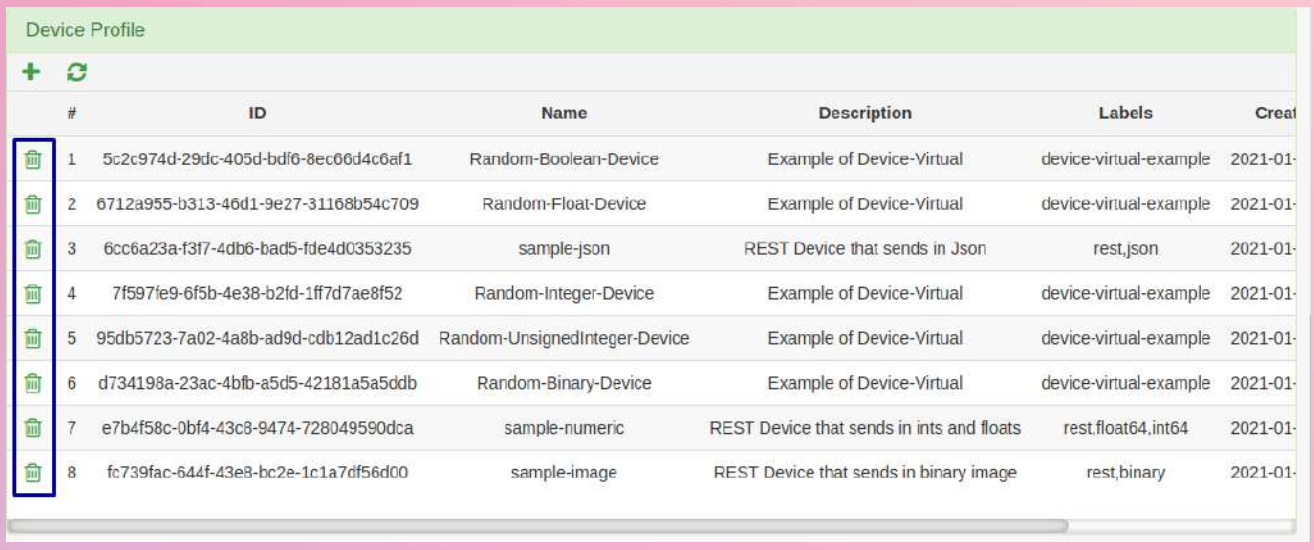
The "OperatingState" should be "ENABLED" and the "AdminState" should be "UNLOCKED".

We decide which device profile to associate the device with in the "DeviceProfileName" section.

We can add the device information in the "DeviceAddressable" section. The information here can be increased or decreased if the "other" option is selected. Some optional information can also be reduced.

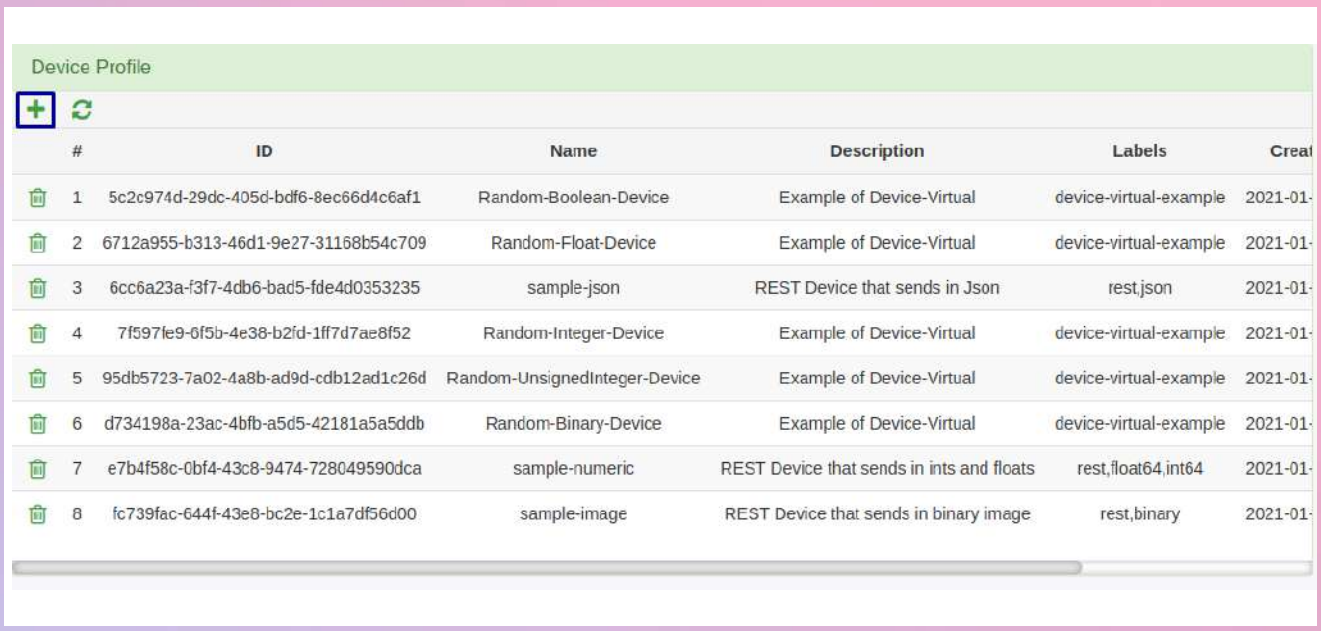


In the "Device Profile" window, we can create or remove device profiles. We can use the trash bin icon to remove the device profile.



#	ID	Name	Description	Labels	Creat
1	5c2c974d-29dc-405d-bdf6-8ec66d4c6af1	Random-Boolean-Device	Example of Device-Virtual	device-virtual-example	2021-01-
2	6712a955-b313-46d1-9e27-31168b54c709	Random-Float-Device	Example of Device-Virtual	device-virtual-example	2021-01-
3	6cc6a23a-f3f7-4db6-bad5-fde4d0353235	sample-json	REST Device that sends in Json	rest,json	2021-01-
4	7f597fe9-6f5b-4e38-b2fd-1ff7d7ae8f52	Random-Integer-Device	Example of Device-Virtual	device-virtual-example	2021-01-
5	95db5723-7a02-4a8b-ad9d-cdb12ad1c26d	Random-UnsignedInteger-Device	Example of Device-Virtual	device-virtual-example	2021-01-
6	d734198a-23ac-4bfb-a5d5-42181a5a5ddb	Random-Binary-Device	Example of Device-Virtual	device-virtual-example	2021-01-
7	e7b4f58c-0bf4-43c8-9474-728049590dca	sample-numeric	REST Device that sends in ints and floats	rest,float64,int64	2021-01-
8	fc739fac-644f-43e8-bc2e-1c1a7df56d00	sample-image	REST Device that sends in binary image	rest,binary	2021-01-

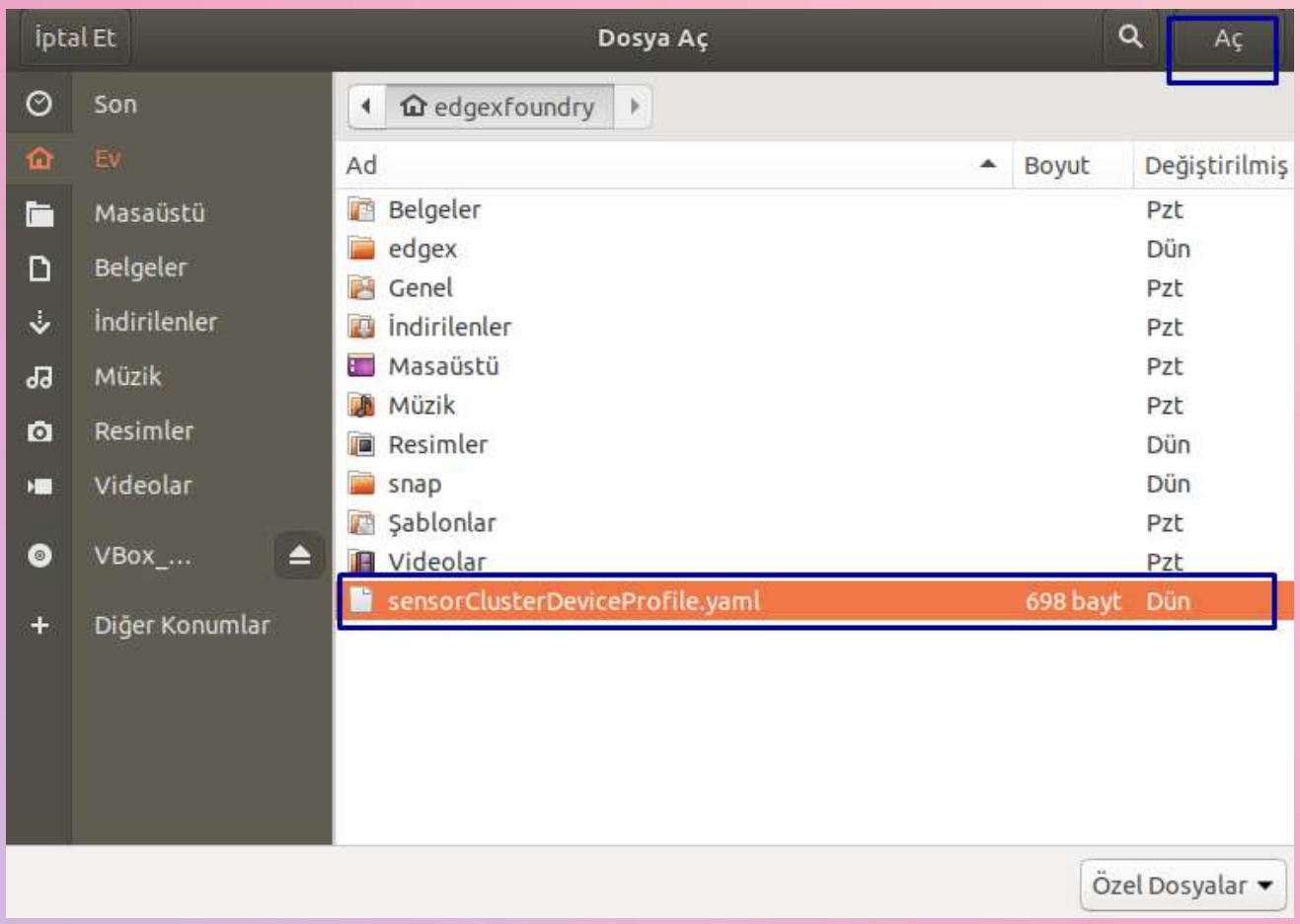
To add a device profile, we need to click the "+" icon.



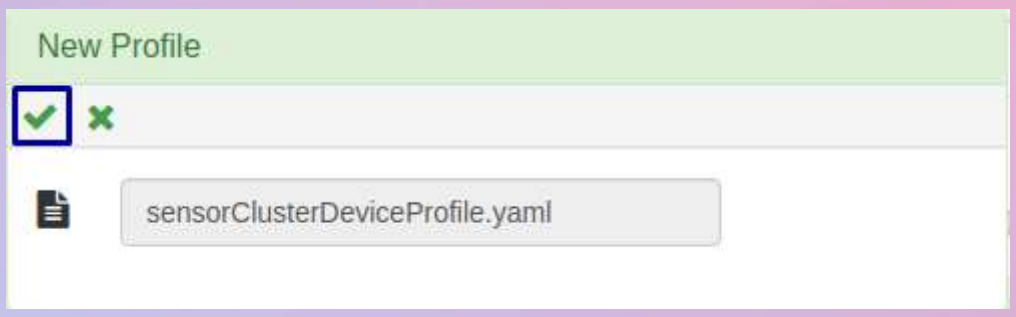
#	ID	Name	Description	Labels	Creat
1	5c2c974d-29dc-405d-bdf6-8ec66d4c6af1	Random-Boolean-Device	Example of Device-Virtual	device-virtual-example	2021-01-
2	6712a955-b313-46d1-9e27-31168b54c709	Random-Float-Device	Example of Device-Virtual	device-virtual-example	2021-01-
3	6cc6a23a-f3f7-4db6-bad5-fde4d0353235	sample-json	REST Device that sends in Json	rest,json	2021-01-
4	7f597fe9-6f5b-4e38-b2fd-1ff7d7ae8f52	Random-Integer-Device	Example of Device-Virtual	device-virtual-example	2021-01-
5	95db5723-7a02-4a8b-ad9d-cdb12ad1c26d	Random-UnsignedInteger-Device	Example of Device-Virtual	device-virtual-example	2021-01-
6	d734198a-23ac-4bfb-a5d5-42181a5a5ddb	Random-Binary-Device	Example of Device-Virtual	device-virtual-example	2021-01-
7	e7b4f58c-0bf4-43c8-9474-728049590dca	sample-numeric	REST Device that sends in ints and floats	rest,float64,int64	2021-01-
8	fc739fac-644f-43e8-bc2e-1c1a7df56d00	sample-image	REST Device that sends in binary image	rest,binary	2021-01-

When clicking the "+" icon, a window like the one below opens.

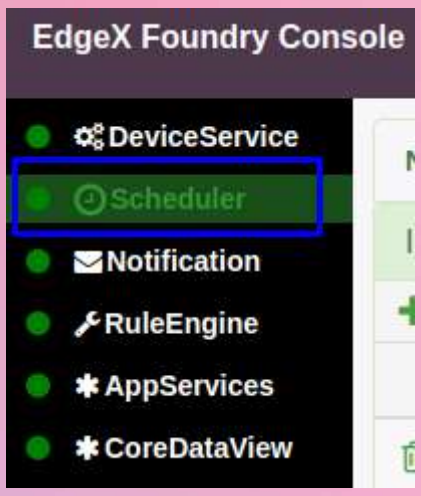
Here, we can click the "file" icon and select the ".yml" file that contains the device profile information. After selecting the ".yml" file and clicking the "Open" button, this window closes and the name of the ".yml" file we selected will appear in the blank space in the "New Profile" window.



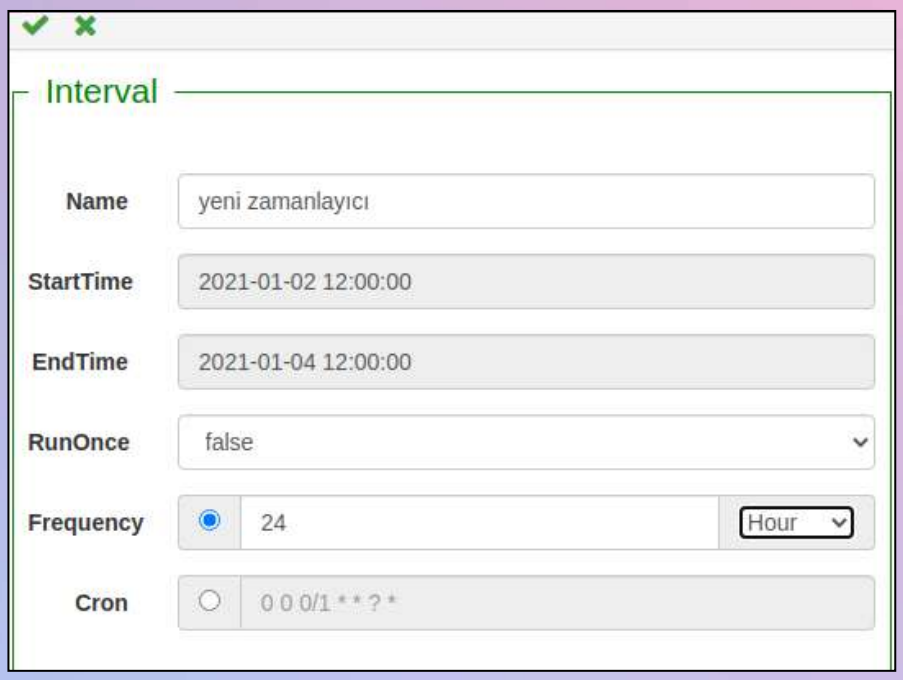
Then, we can add our device profile by clicking the "✔" button.



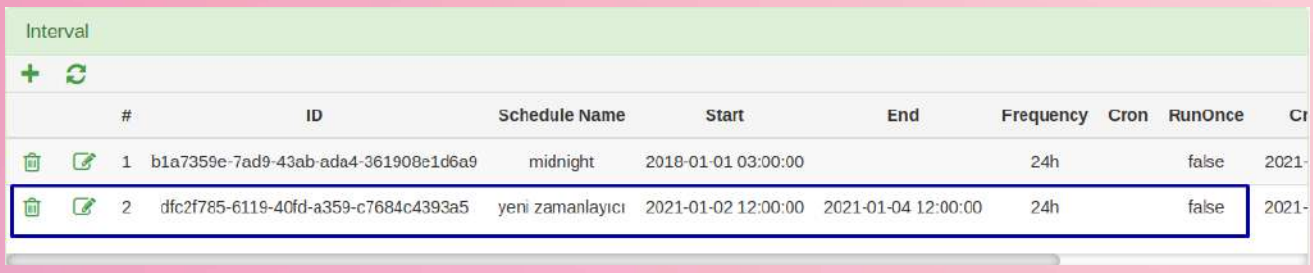
SCHEDULER



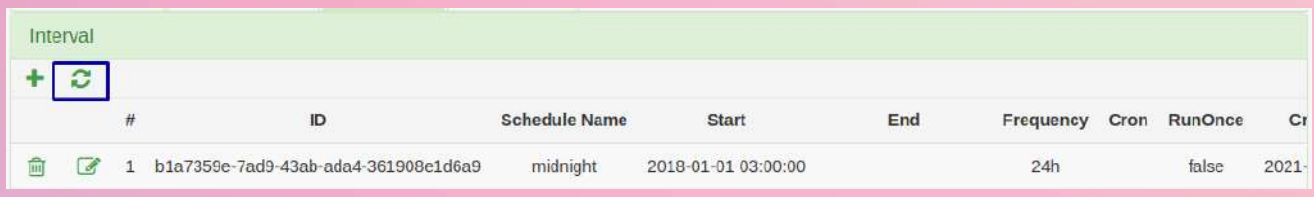
The Scheduler manages all Intervals. In the Scheduler tab, you can add a new interval in the Interval table and specify its start time, end time, whether it will run once, its frequency, and its time-based scheduling (cron) operations. To add a new interval, first click on the "+" icon.



Once you have selected your options, you can create your interval by clicking on the "✔" button.

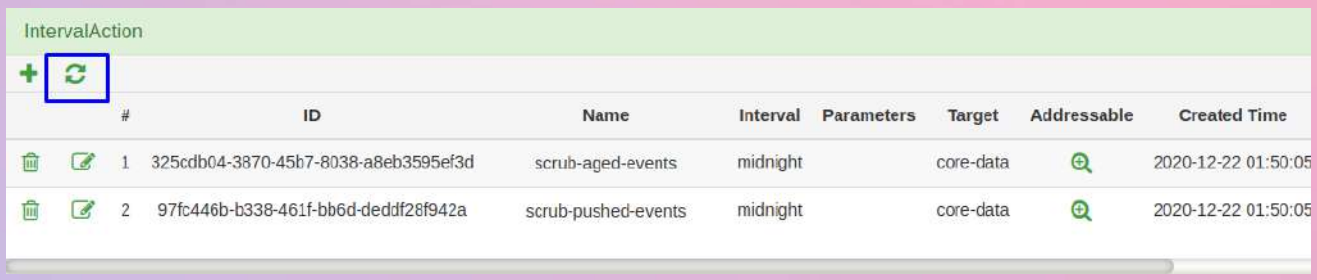


The interval you added will be listed like this. To refresh this list, you can use the refresh icon next to the "+" button.



If you create a new interval, you also need to create actions for it. You can do this using the IntervalAction table.

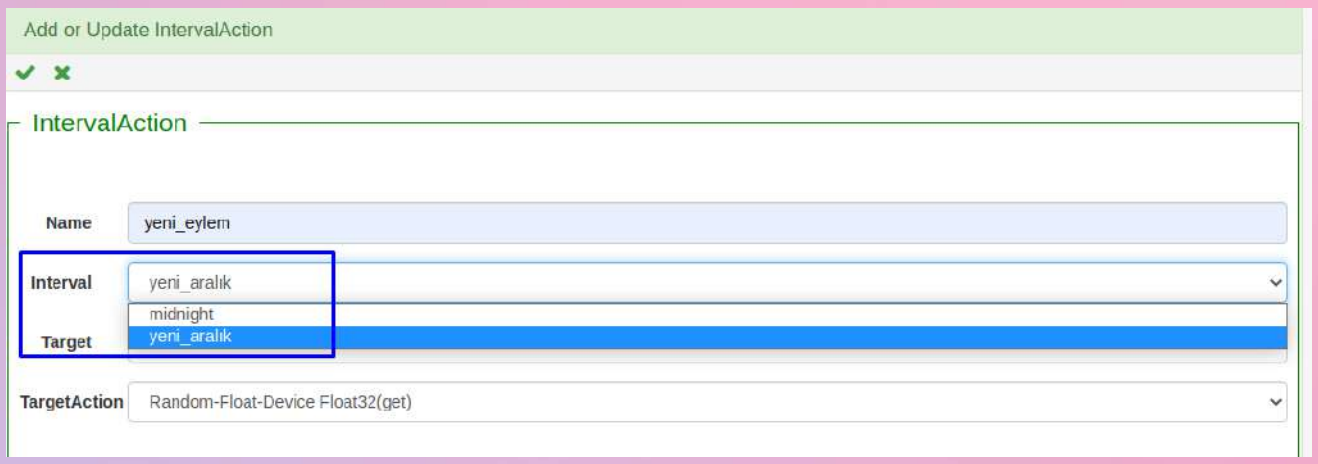
You can refresh the list using the refresh icon next to the "+" icon as standard.



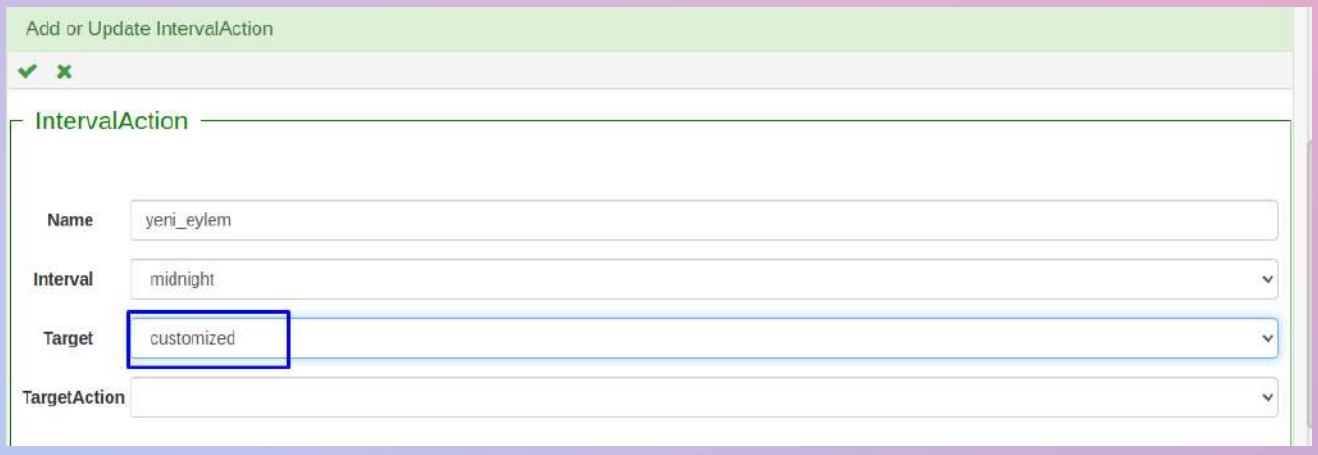
The "Addressable" button in the IntervalAction table gives us method and URL information.

After creating a new interval, we said that we will use the IntervalAction table to perform an action for it. We will do this using the "+" icon in the IntervalAction table.

After entering the name we want for this action in the "Name" tab, in the "Interval" tab, the interval names we created in the Interval table above will appear and we can create an action for the desired interval.



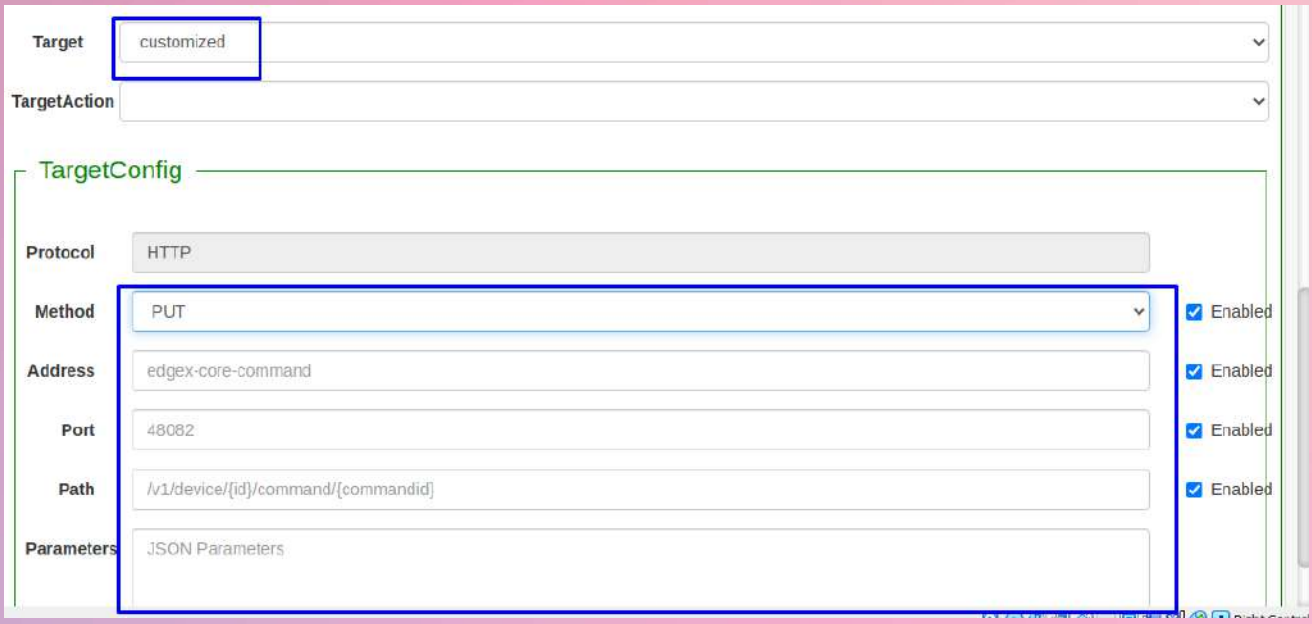
If we change our selection in the Target tab to "customized", we can create a customized interval action.



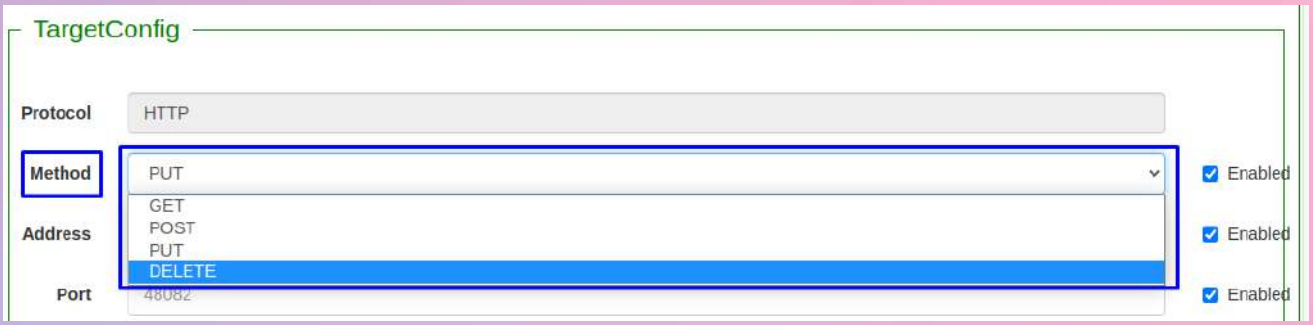
If we select "core-command" instead of "customized" in the Target tab, we can create actions for the devices that appear.

After selecting the device, the "TargetConfig" tab displays automatically generated information for the action. To activate this information, we can use the "Enabled" buttons next to the tab.

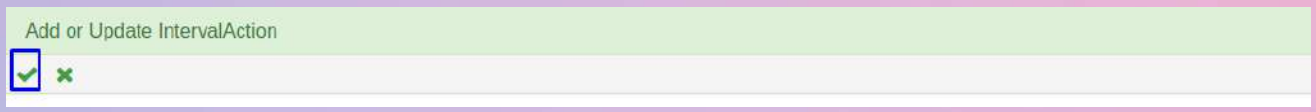
As I mentioned earlier, if we change the "Target" tab in the IntervalAction table to "customized", the "TargetConfig" table will come as a customizable table. Valid values for this must be entered here.



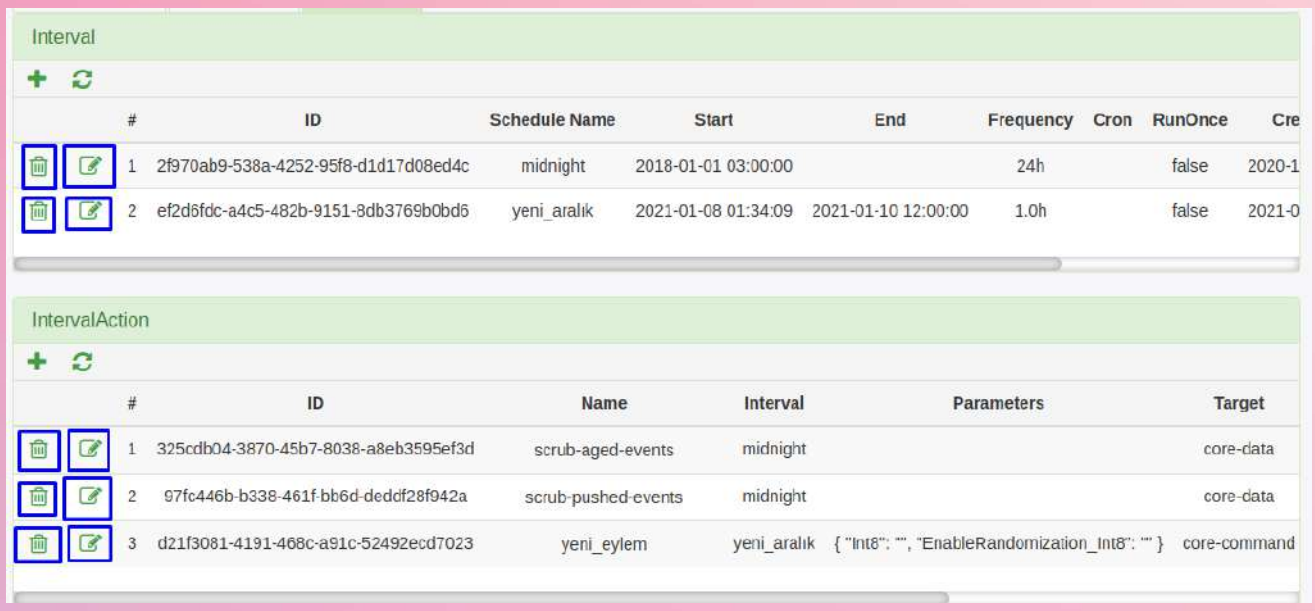
If you have changed the "Target" tab to "customized", you can determine the method in the "Method" tab of the "TargetConfig" table based on the operation you want to perform.



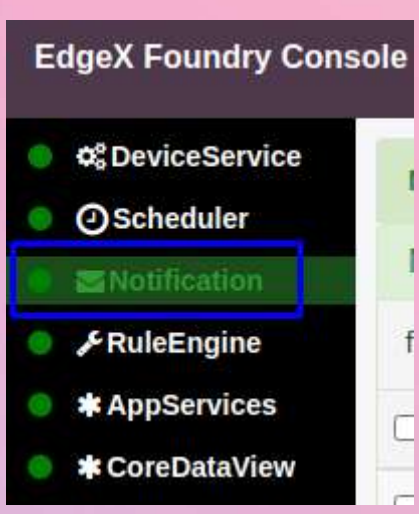
After completing your operations, you can add your interval action by clicking the "✓" button.



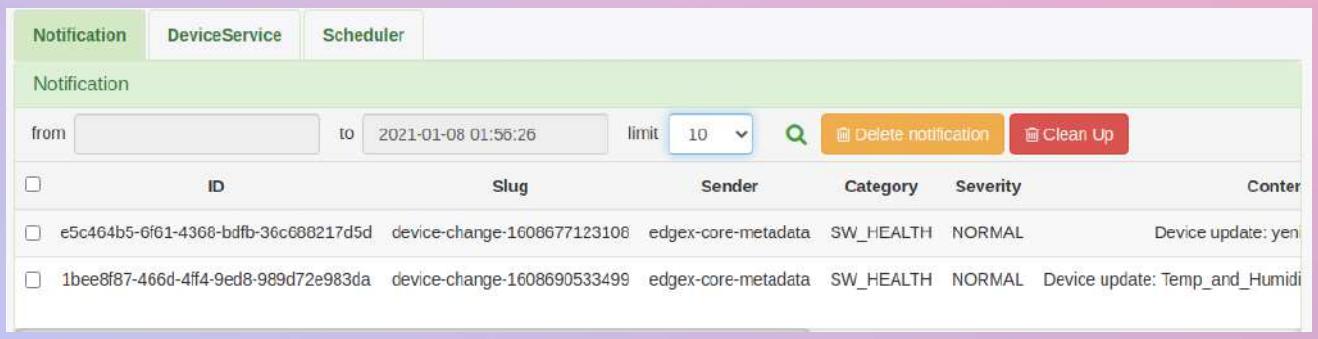
You can use the trash can and edit icon to edit and delete the intervals and actions that will be applicable for these intervals that you have added.



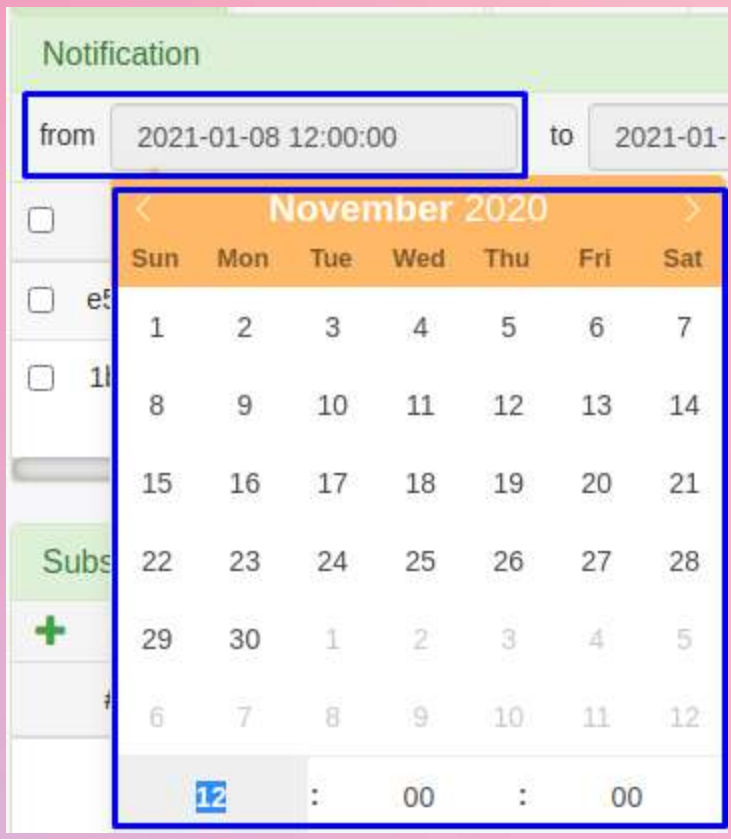
NOTIFICATION



In the "Notification" screen, you can manage notifications, clear notifications, create a subscription to receive notifications, and view transmissions.

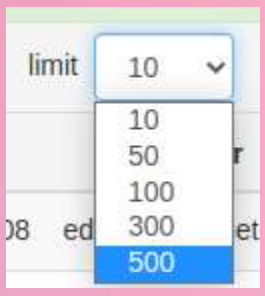


The notifications appear in the "Notification" table and the "from"-"to" sections are used to receive notifications at specific time intervals, and by clicking these sections, a calendar is displayed. By clicking the space next to the "from" section, you can select the start time.

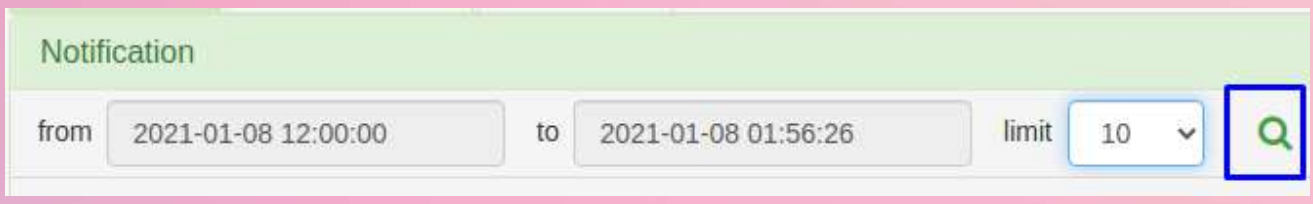


By clicking the space next to the "to" section, you can select the end time.

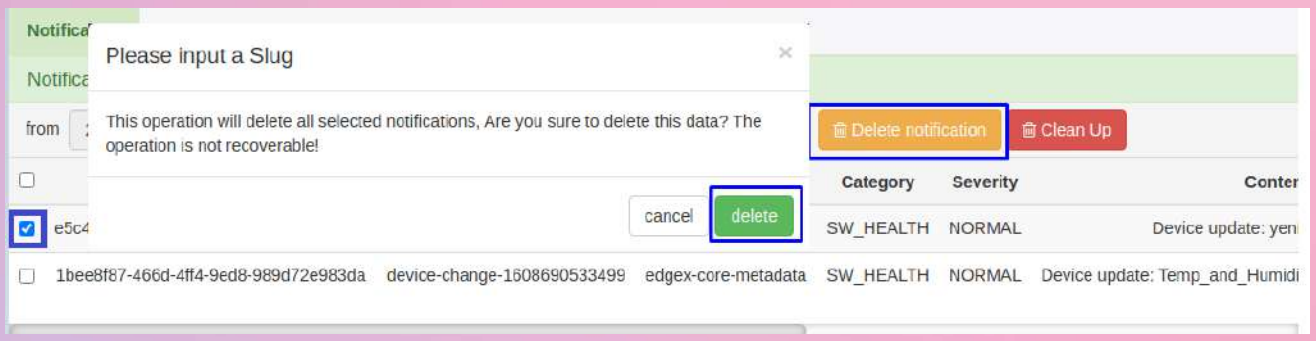
From the "limit" section, you can select how many notifications will be displayed.



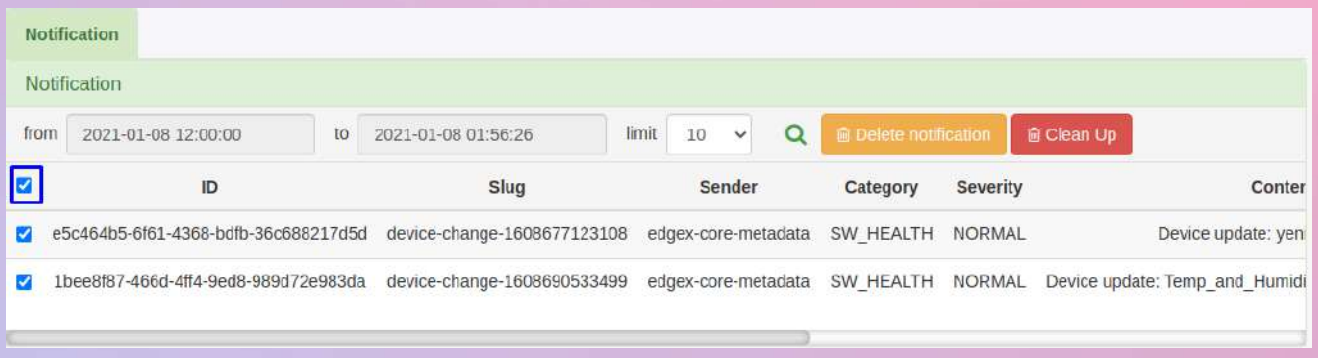
The magnifying glass icon displays notifications within the specified range and amount.



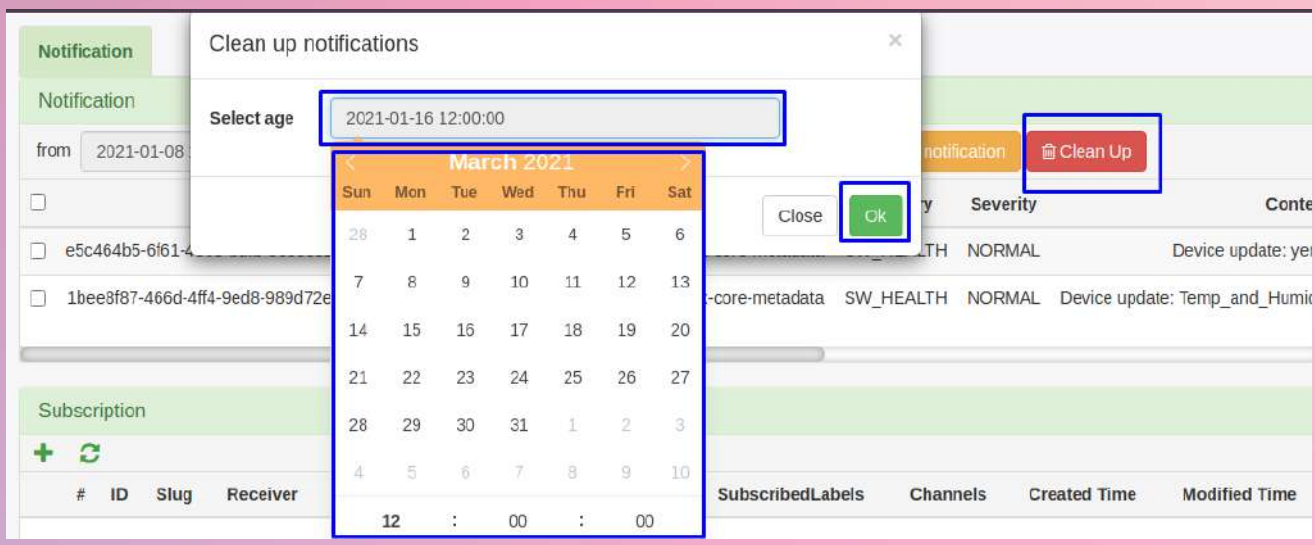
If you want to delete the notifications you have selected, you need to mark the blank space on the left side of the notification you want to delete and then click the "Delete Notification" button and confirm it in the notification screen that appears.



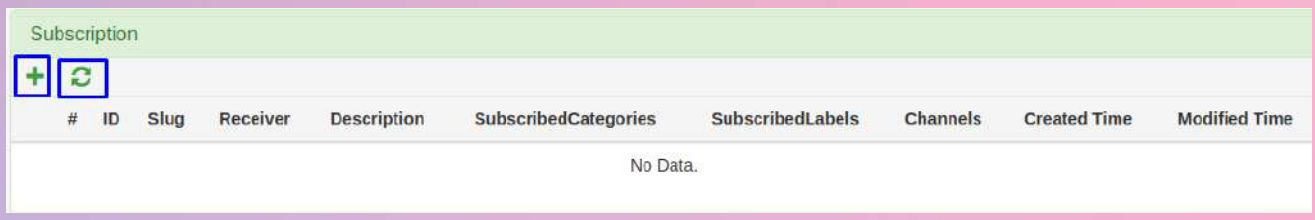
If you want to delete all notifications at once, you need to mark the blank space immediately above the marking section.



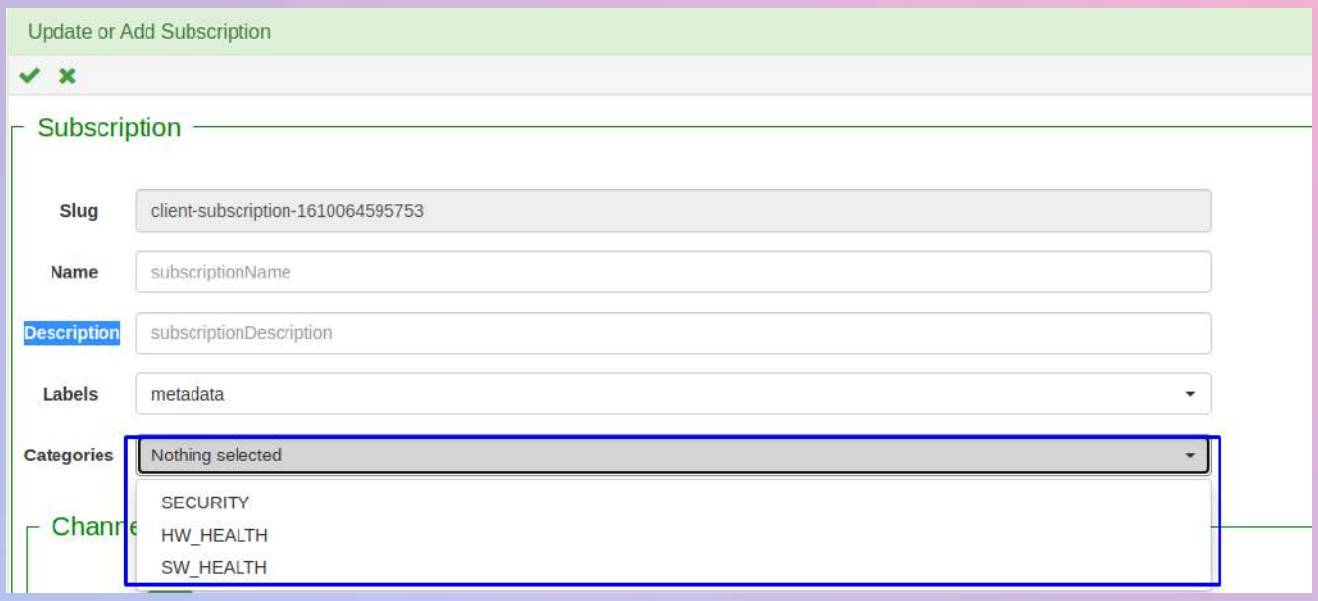
If you want to delete notifications on a specific date, you need to click the "Clean Up" button and select the date.



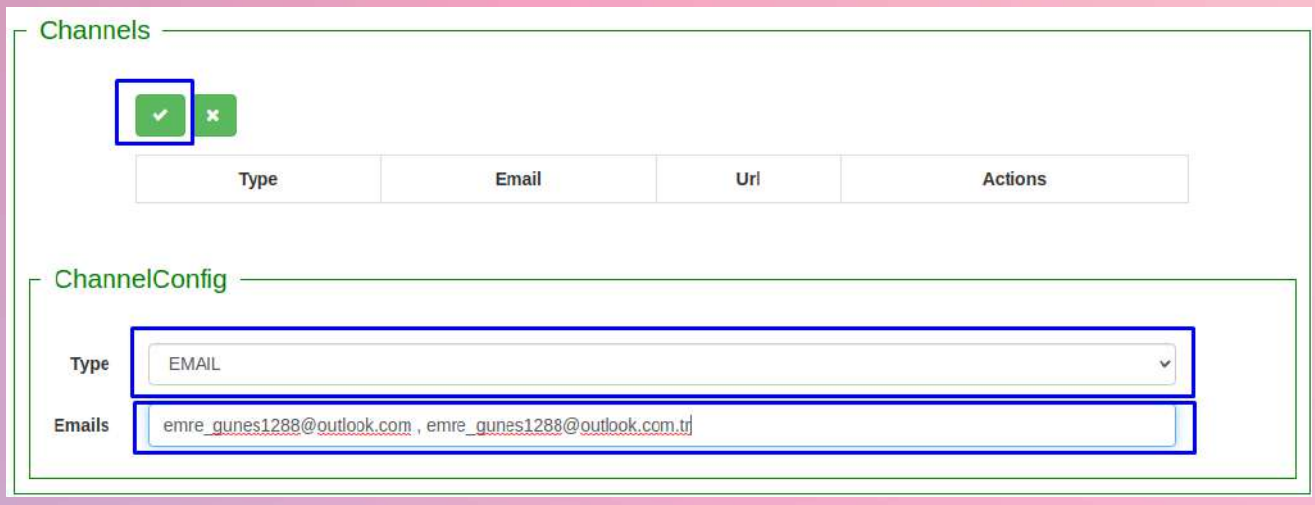
In the "Subscribe" table, you can create a subscription to receive notifications and configure it. You can create a new subscription by clicking the "+" button, and you can refresh the table by clicking the "refresh" button.



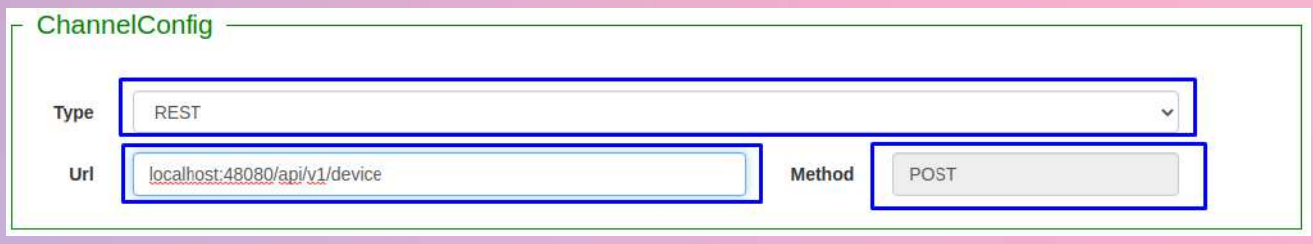
A unique identifier is created for you in the "Slug" section. After adding the "Name" and "Description" according to your own request, you can specify what type of notifications you want to receive in the "Categories" section. The type of the content of the notifications you will receive is found in the "Labels" section and only the "metadata" option is available.



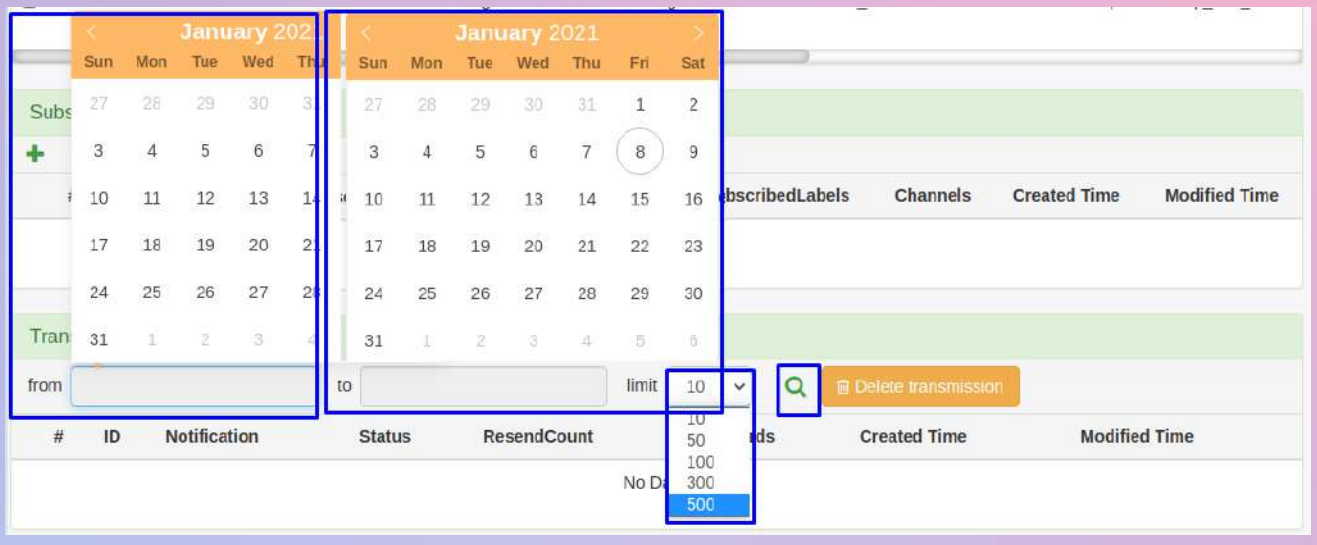
In the "Channel" table at the bottom, you can create a channel and receive notifications to your email or web address. If you want to receive them as email, click the "+" button and in the "ChannelConfig" table that appears, set the "Type" section to "E-MAIL" and enter your email. If you want to receive notifications to multiple emails, you can separate them with a comma.



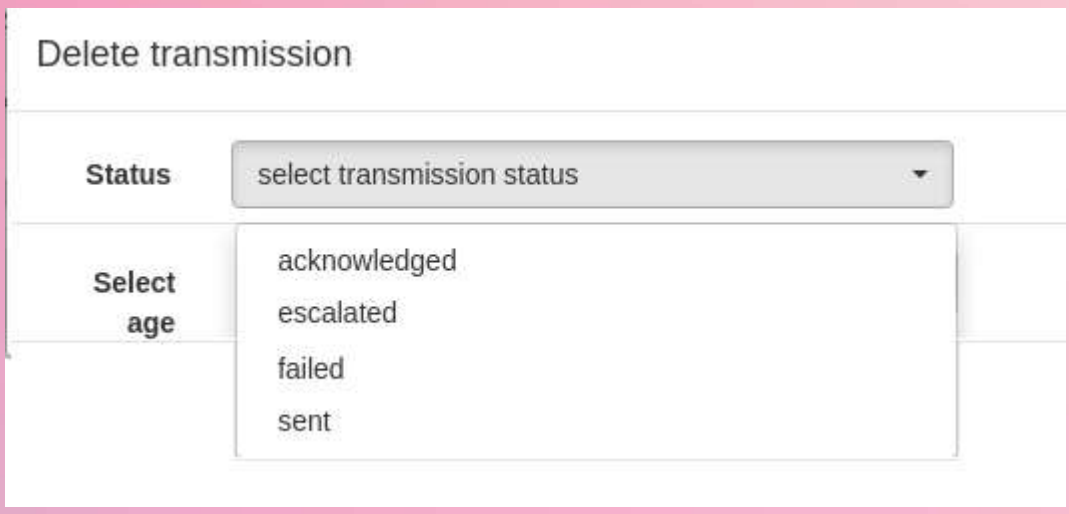
If you want to receive notifications from a URL, you need to change the "Type" to "REST" and enter a valid URL address. The "Method" will remain fixed as "POST" for notifications to be sent.



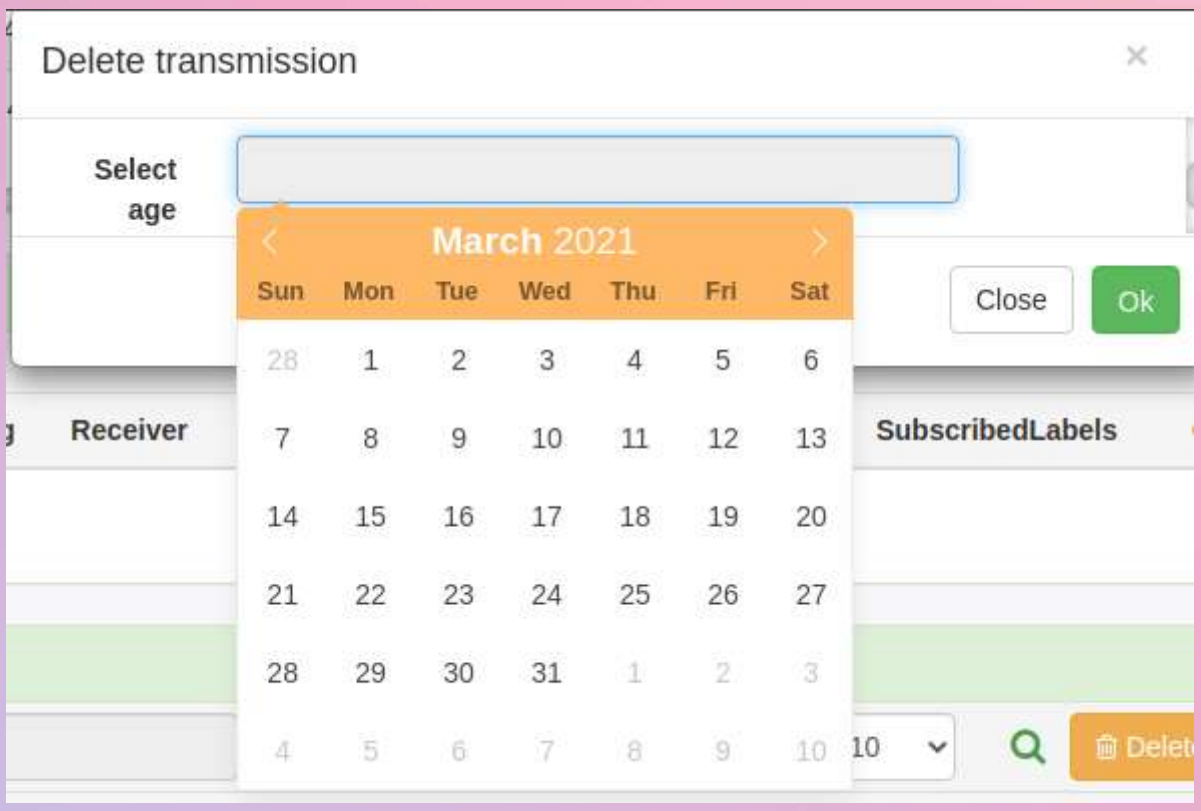
In the "Transmission" table, the transmitted notifications are displayed. You can filter by start time from the "from" section, end time from the "to" section, and number of transmissions from the "limit" section. Then, the filtering process is performed by clicking the "magnifying glass" button and searching.



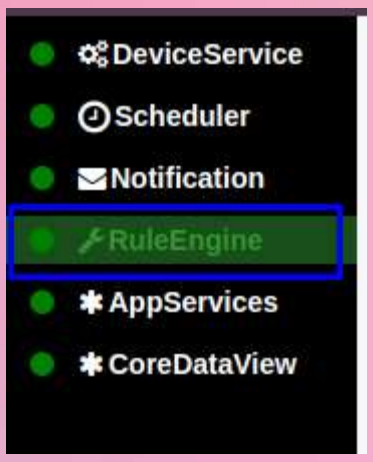
You can filter accepted, sent, failed, and elevated notifications and delete them by clicking the "Delete transmission" button. Multiple selections can also be made.



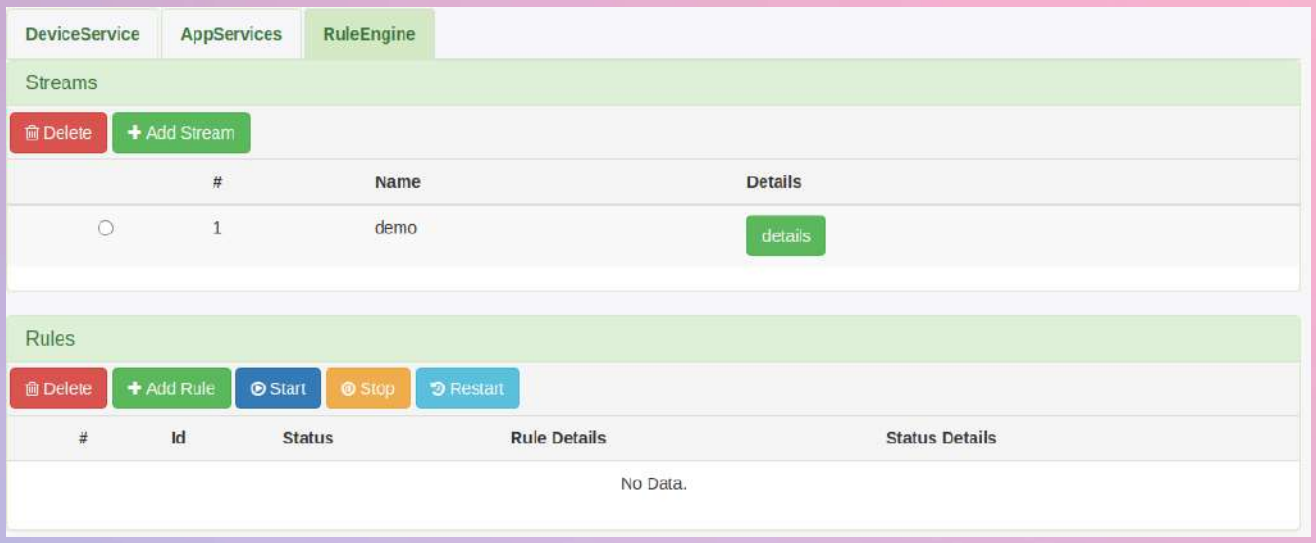
Notifications filtered for deletion can also be deleted based on specific dates in the "Select age" tab.



RULE ENGINE

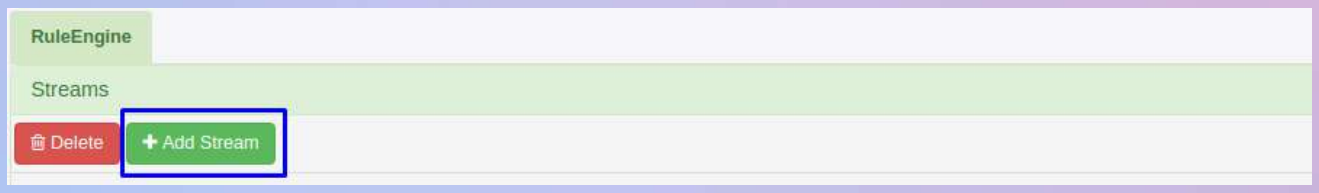


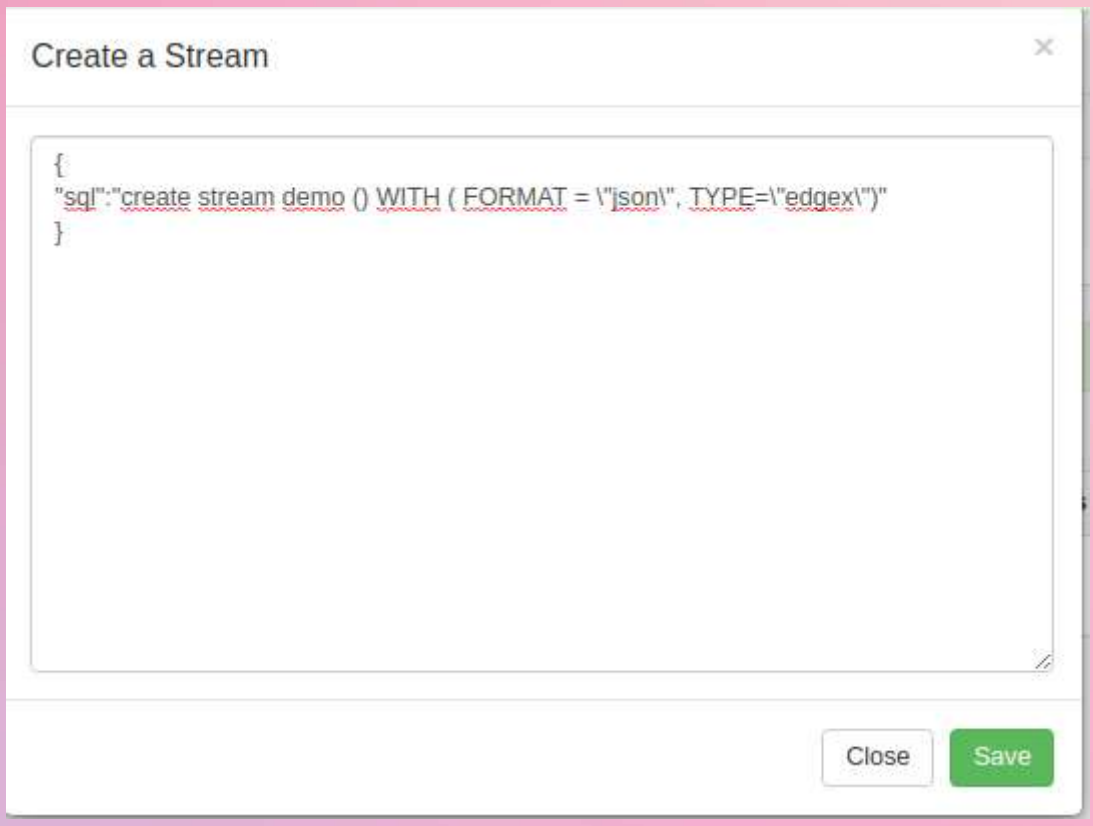
In the "RuleEngine" tab, streams and rules can be created.



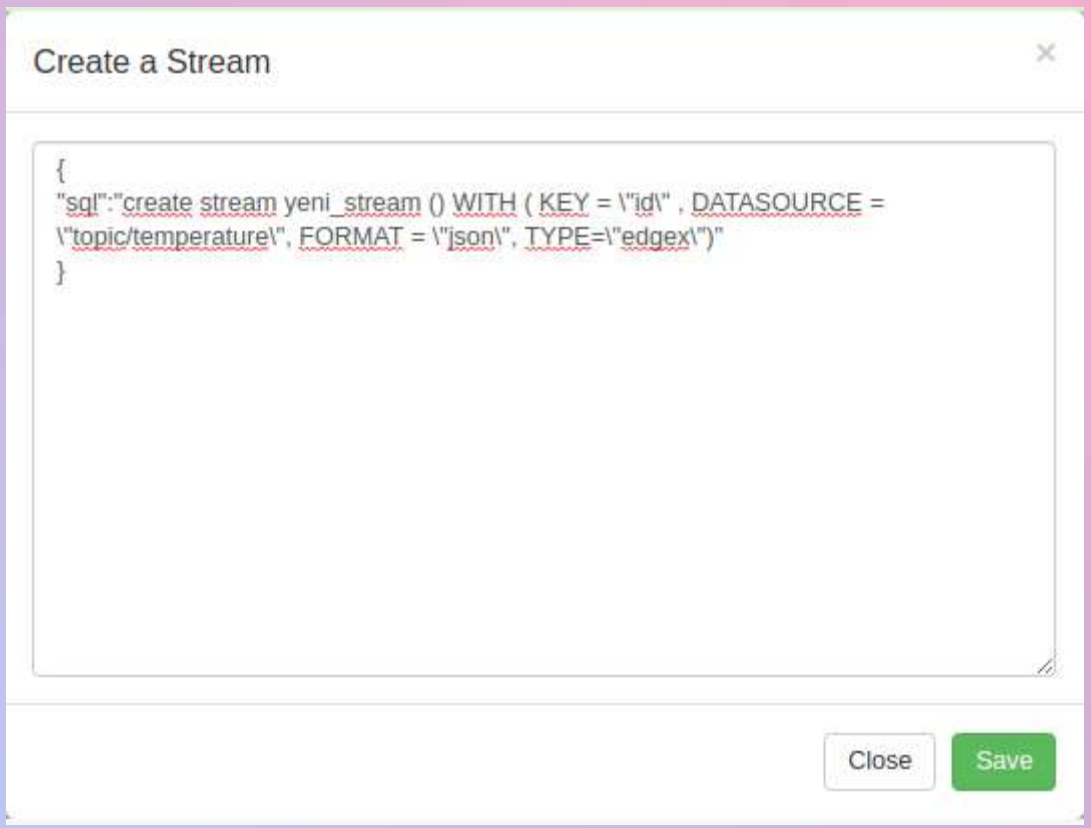
When adding a new stream, you can click the "+ Add Stream" button and add some optional features in the window that appears. These can be listed as follows:

-DATASOURCE
-FORMAT
-KEY
-TYPE
-StrictValidation
-CONF_KEY

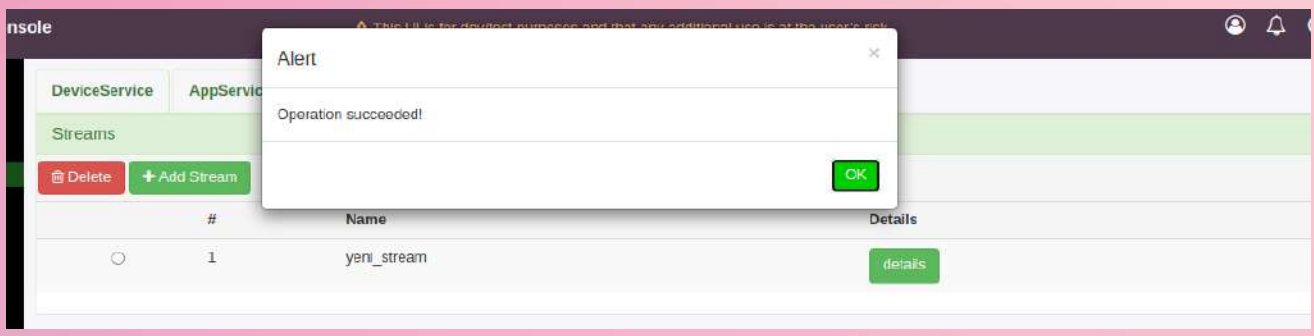




In this example, "demo" is the stream name. You can write the desired parameters inside the "...". For example;



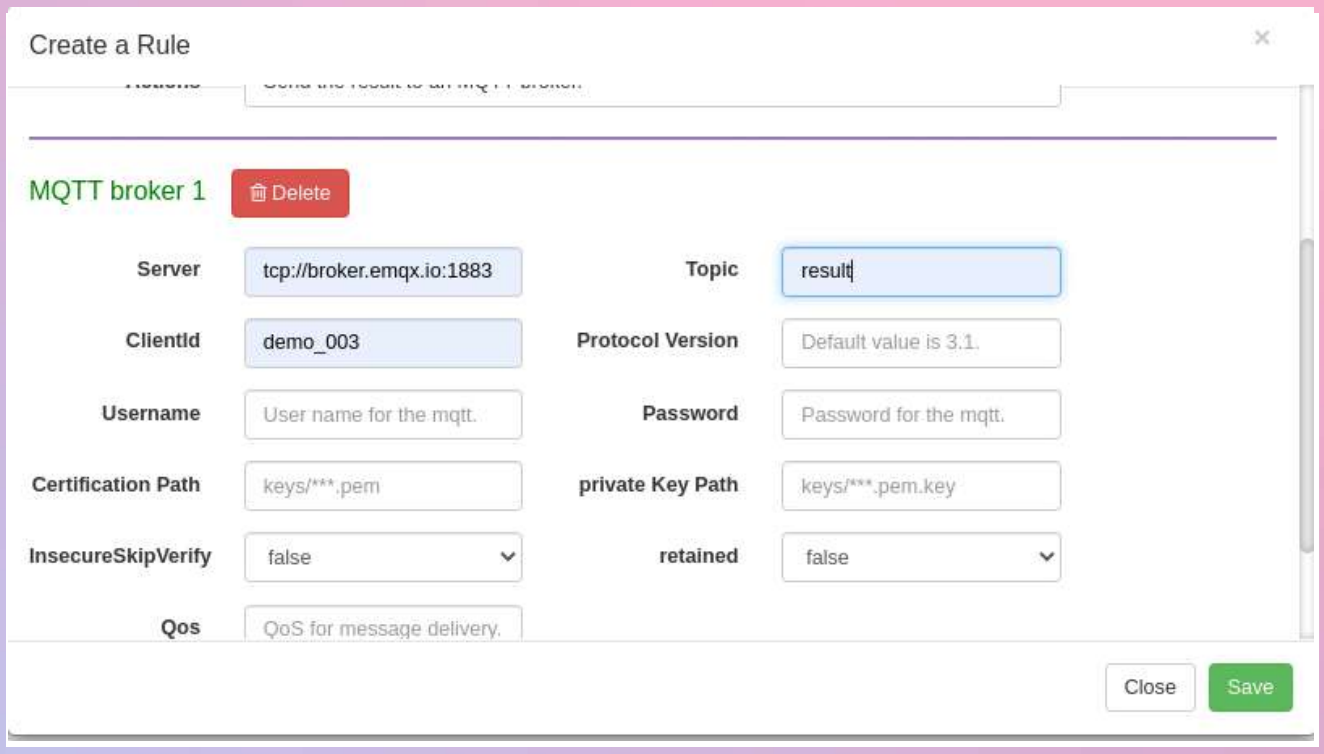
If you enter correctly and correctly, you will receive the "Operation succeeded!" notification.



The added stream will be displayed in the "Streams" table and the entries added with the "details" button will show "json" or "binary" output.

The operation is performed in the Result table to send the result data based on the action you selected. When you click the "+ Add Rule" button, you need to enter a unique identifier in the "ID" section. In the "Action" section, you must specify where to send the result data. Here, multiple selections can be made and windows are created for each selection.

Let's create a rule that sends result data to an MQTT broker.

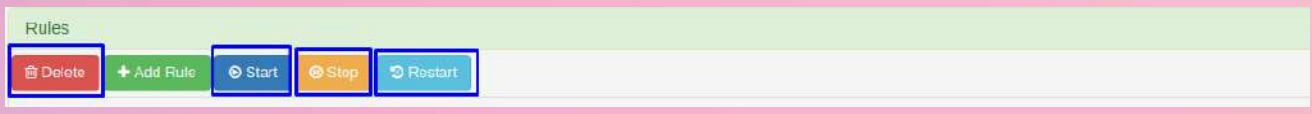


To view the information of the created rule, on our terminal screen:

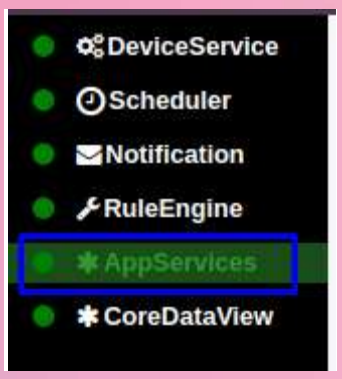
```
docker logs edgex-kuiper
```

```
time="2021-01-08T14:35:00Z" level=info msg="Rule rule12 is created." file="xsql_processor.go:226"
time="2021-01-08T14:35:00Z" level=info msg="Init rule with options {isEventTime: false, lateTolerance: 0, concurrency: 1, bufferLength: 1024"
file="xsql_processor.go:399"
time="2021-01-08T14:35:00Z" level=info msg="Opening stream" file="streams.go:89" rule=rule12
time="2021-01-08T14:35:00Z" level=info msg="open source node demo with option map[FORMAT:txt TYPE:edgex]" file="source_node.go:59" rule=rule12
time="2021-01-08T14:35:00Z" level=info msg="open sink node 1 instances" file="sink_node.go:143" rule=rule12
time="2021-01-08T14:35:00Z" level=info msg="Opening mqtt sink for rule rule12." file="mqtt_sink.go:113" rule=rule12
time="2021-01-08T14:35:00Z" level=info msg="Connect MQTT broker with username and password." file="mqtt_sink.go:132" rule=rule12
time="2021-01-08T14:35:00Z" level=info msg="open source node 1 instances" file="source_node.go:78" rule=rule12
time="2021-01-08T14:35:00Z" level=info msg="Connect to value descriptor service at: http://edgex-core-data:48080/api/v1/valuedescriptor \n" fi
le="edgex_source.go:59"
time="2021-01-08T14:35:00Z" level=info msg="Use configuration for edgex messagebus {{ 0 } {edgex-app-service-configurable-rules 5566 tcp} zero
 map[]}\n" file="edgex_source.go:81"
time="2021-01-08T14:35:00Z" level=info msg="Start source demo instance 0 successfully" file="source_node.go:115" rule=rule12
time="2021-01-08T14:35:00Z" level=info msg="The connection to edgex messagebus is established successfully." file="edgex_source.go:115" rule=r
ule12
time="2021-01-08T14:35:00Z" level=info msg="Successfully subscribed to edgex messagebus topic events." file="edgex_source.go:124" rule=rule12
time="2021-01-08T14:35:00Z" level=error msg="Show rules error: Rule 5c2c974d-29dc-405d-bdf0-8ec66d4c6af1 is not found" file="rest.go:42"
time="2021-01-08T14:35:01Z" level=info msg="The connection to server tcp://broker.emqx.io:1883 was established successfully" file="mqtt_sink.g
o:161" rule=rule12
time="2021-01-08T14:35:01Z" level=info msg="Get 28 of value descriptors from service." file="edgex_source.go:312"
edgexfoundry@edgexfoundry-VirtualBox:~$
```

Deleting the created rule is done with the "Delete", starting with "Start", stopping with "Stop", and restarting with "Restart" buttons.



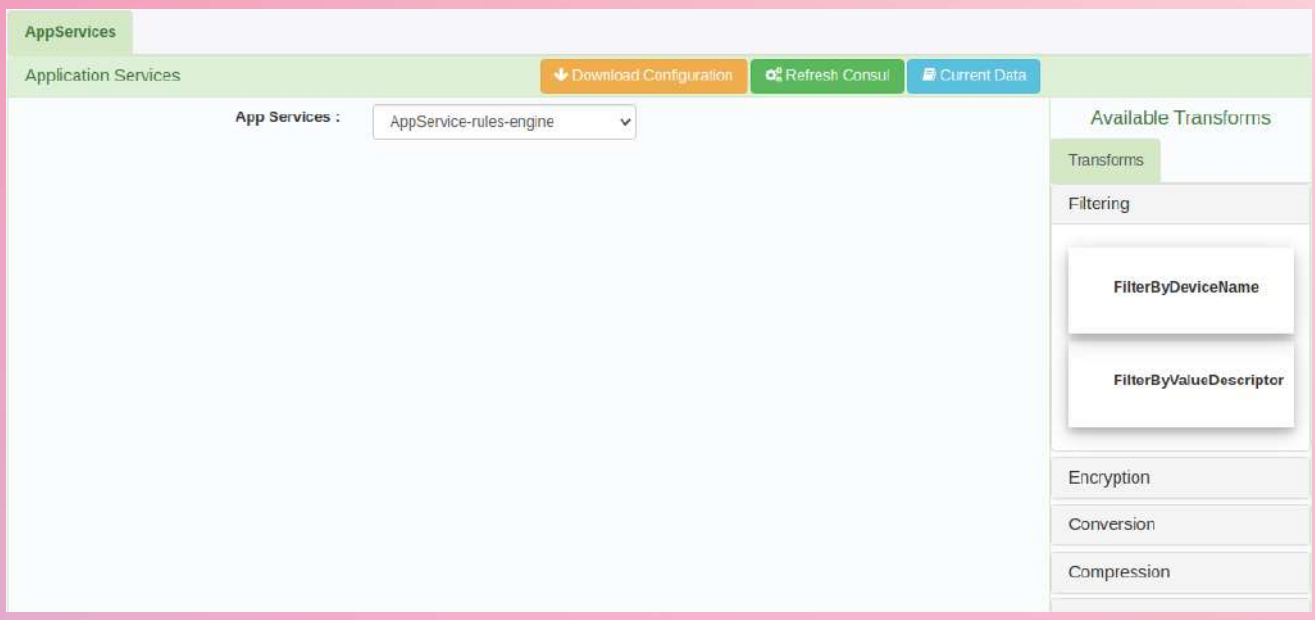
APPSERVICES

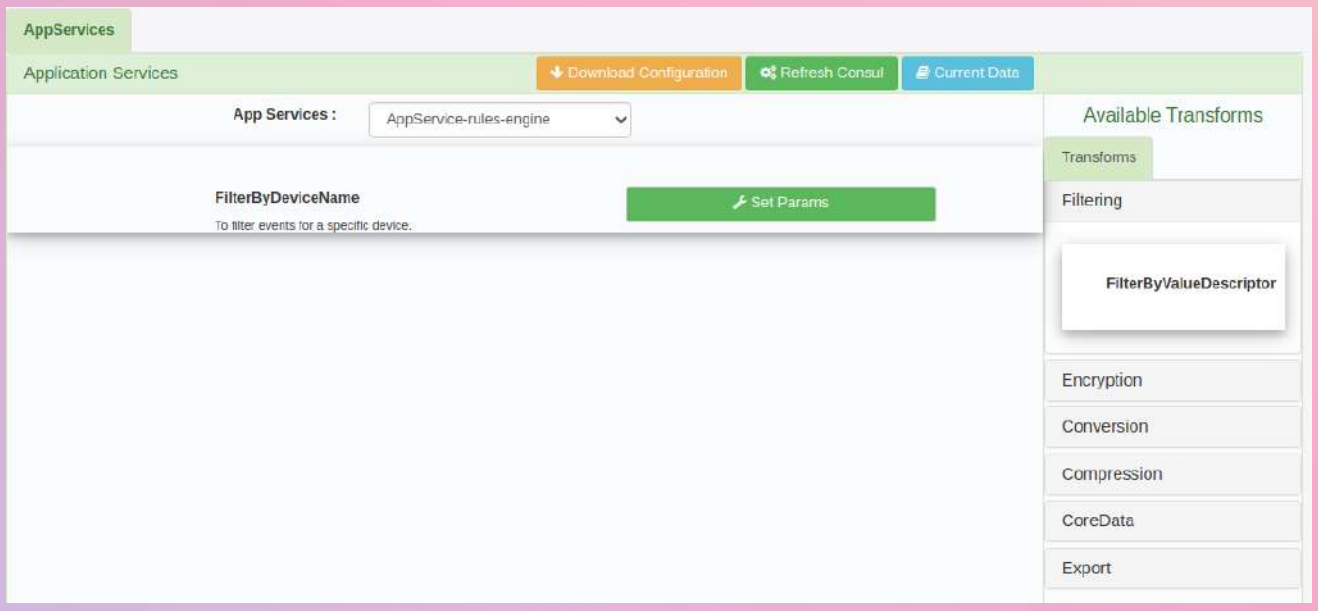


This tab is useful for managing application services. Multiple application services can be managed. All currently supported methods are listed on the right side of the page.

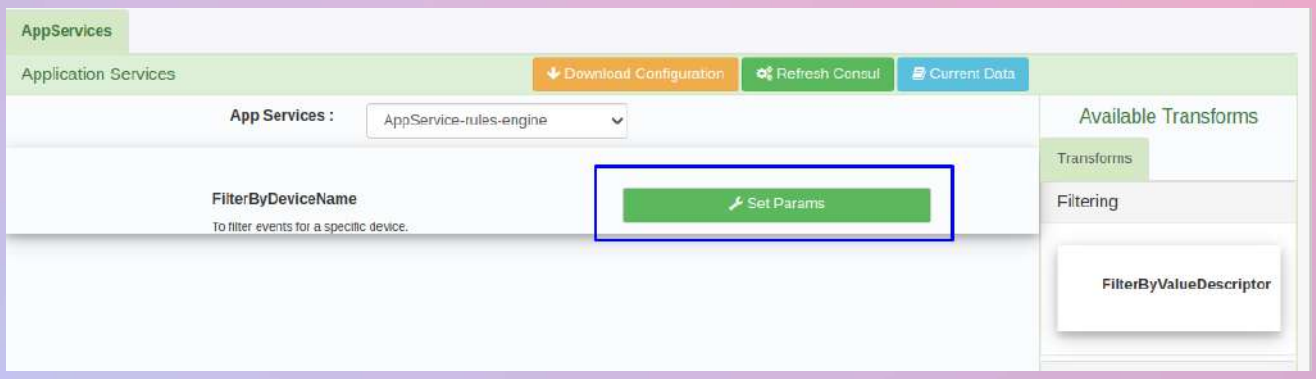
To perform the operation, you must change the "AppServices" tab to "AppService-rules-engine".

To set the parameters of the listed methods on the right, you can drag the desired method with the mouse and drag it to the blank space on the left side of the screen.

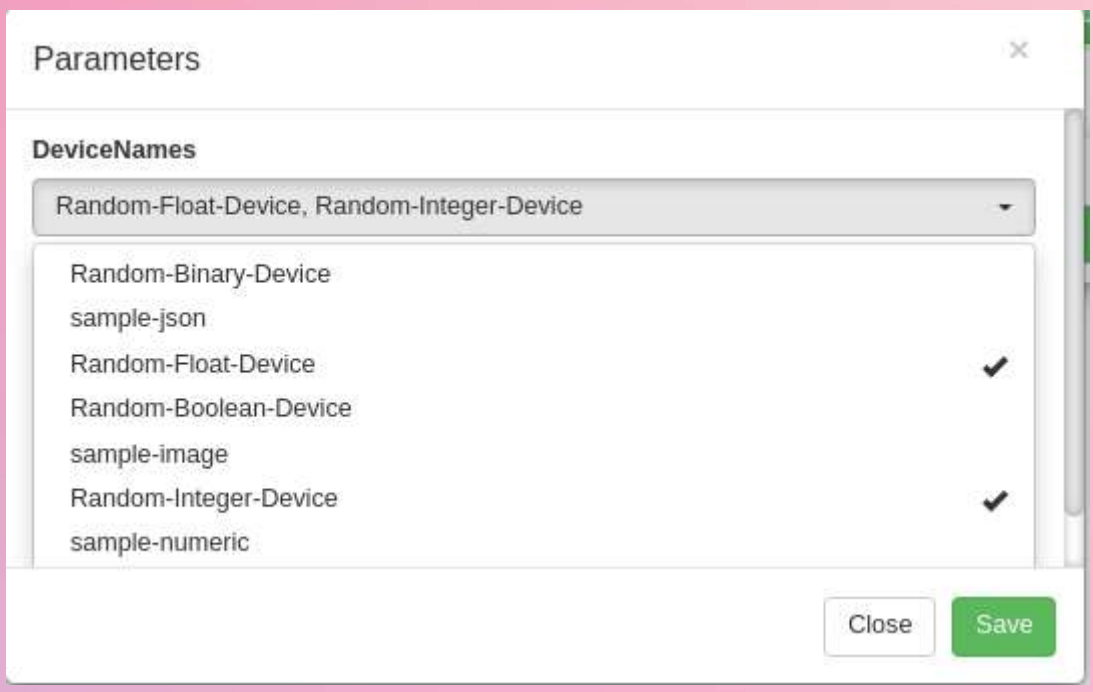




The "Set Params" button should be clicked to set the parameters.



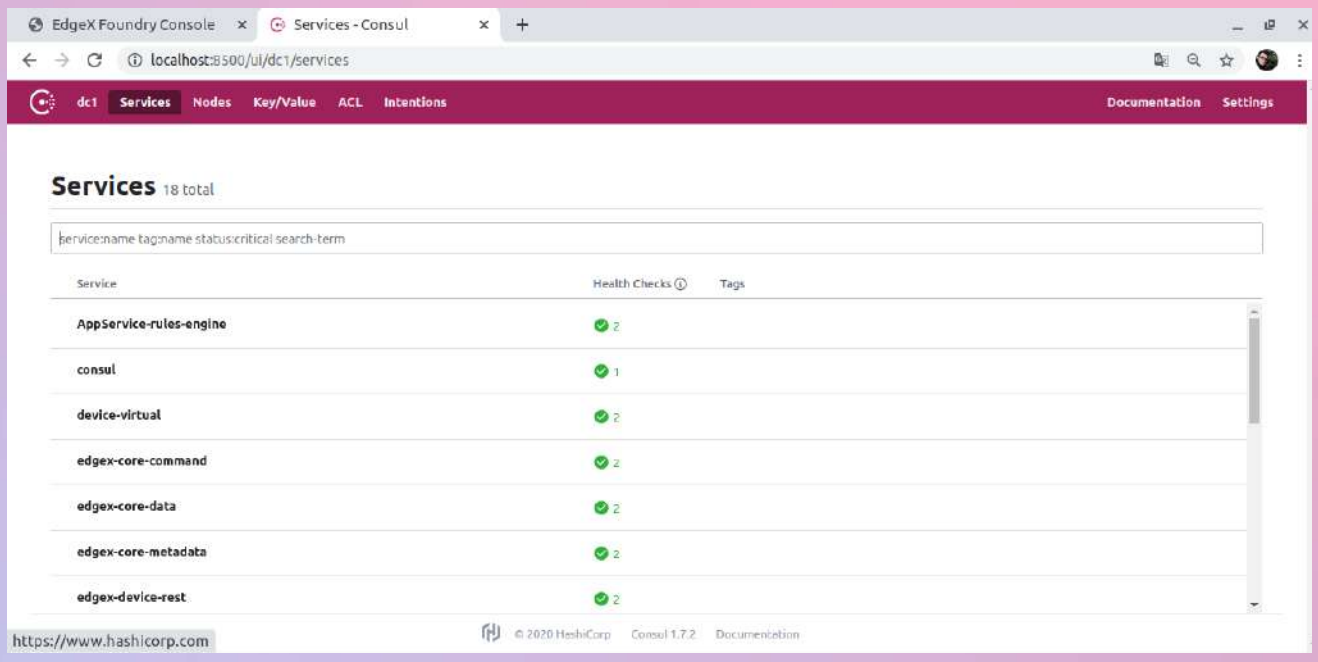
After clicking the "Set Params" button, the desired parameters are specified. In some methods, multiple parameters can be selected.



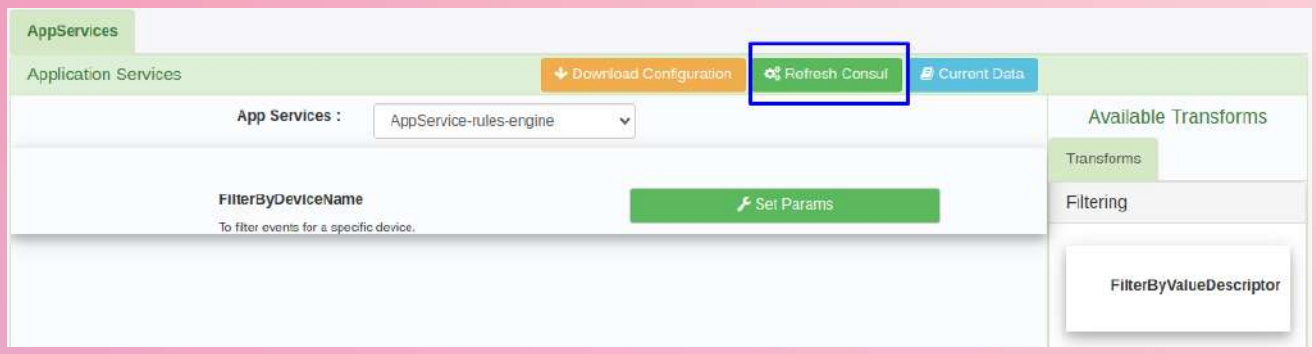
We can view the parameters we distribute in EdgeX through the "Consul" platform from here. To connect to the "Consul" interface, we need to log in to the address in the web browser:

localhost:8500

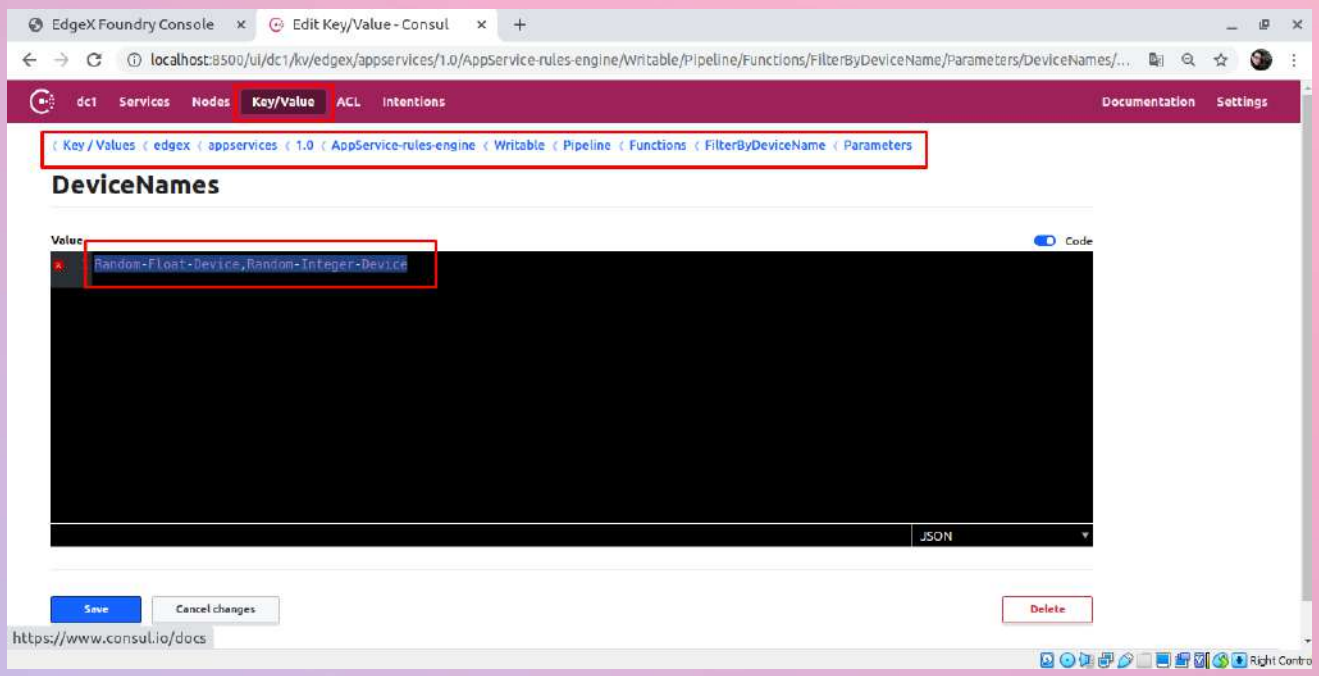
we need to enter the address.



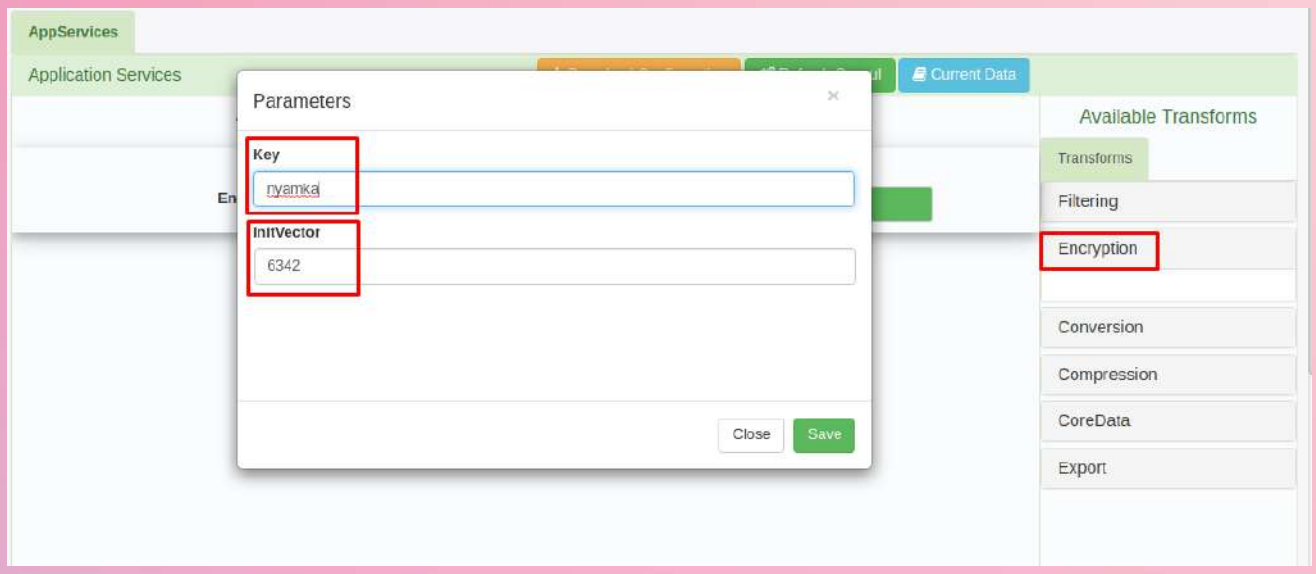
With the display of service information on the "Consul" interface, the parameters sent for the methods in the AppServices of EdgeX are also displayed. However, these parameters must be sent by clicking the "Refresh Consul" button in the EdgeX interface to update the specified parameters in the "Consul" interface.



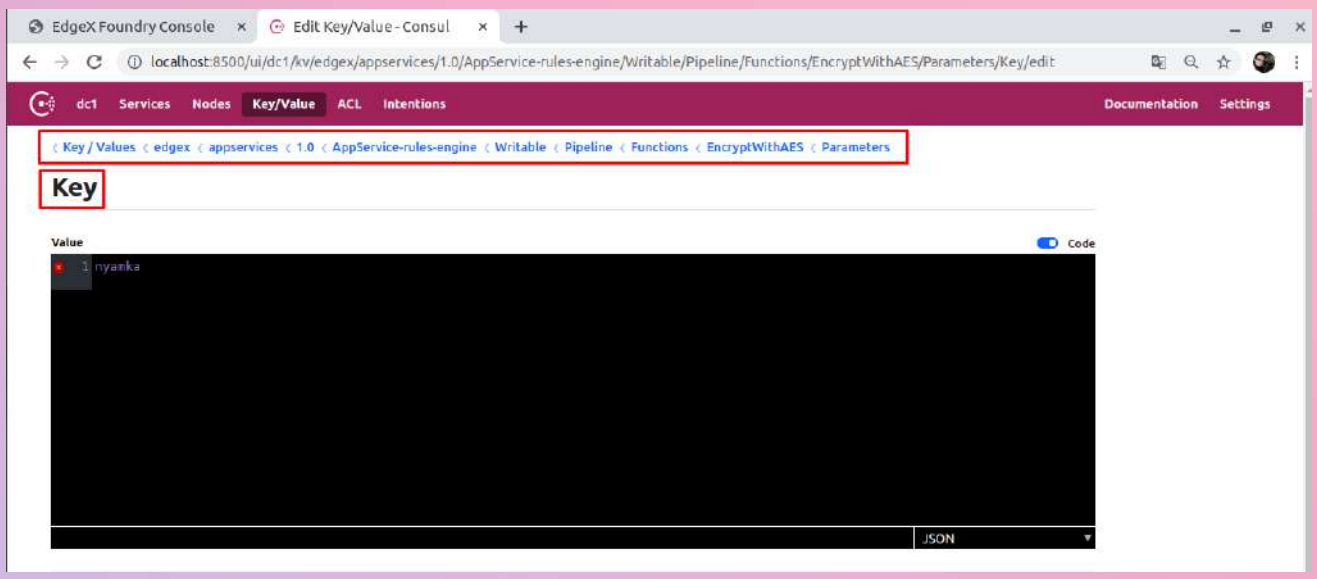
The updated parameters can be viewed from the "Key/Value" tab on the Consul interface. For example, the parameters applied for "FilterByDeviceName" are "Random-Integer-Device" and "Random-Float-Device". To view these parameters in the Consul interface, the following path must be followed.



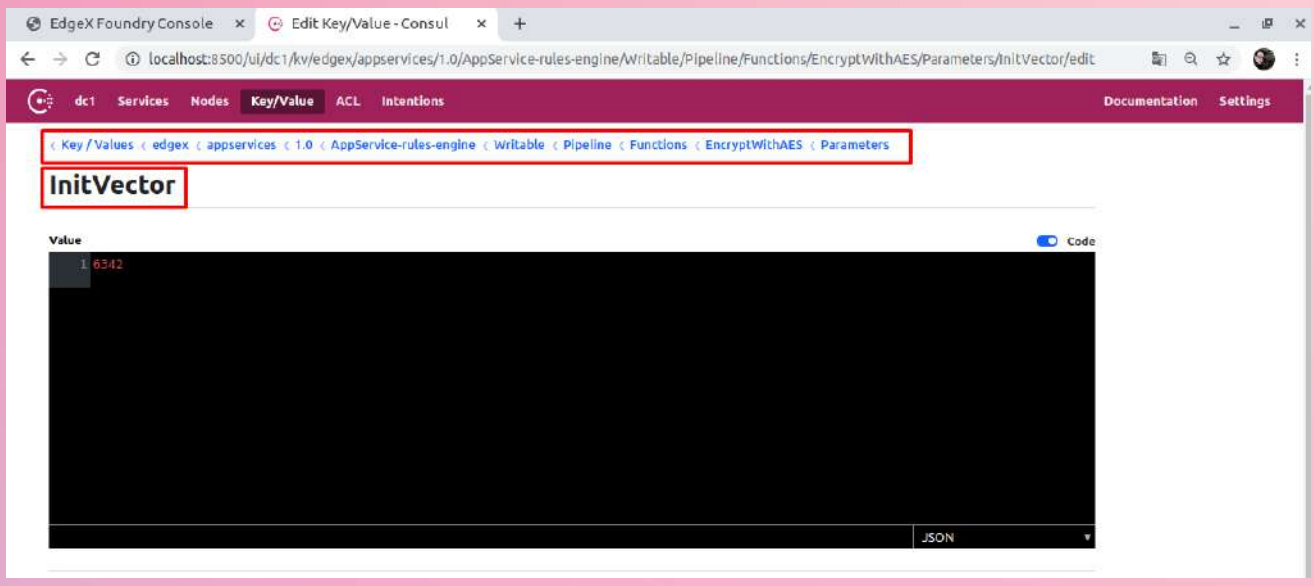
As you can see, we can view the device names selected from "FilterByDeviceName". For another example, let's enter the parameters in the "Encryption" (encryption) service.



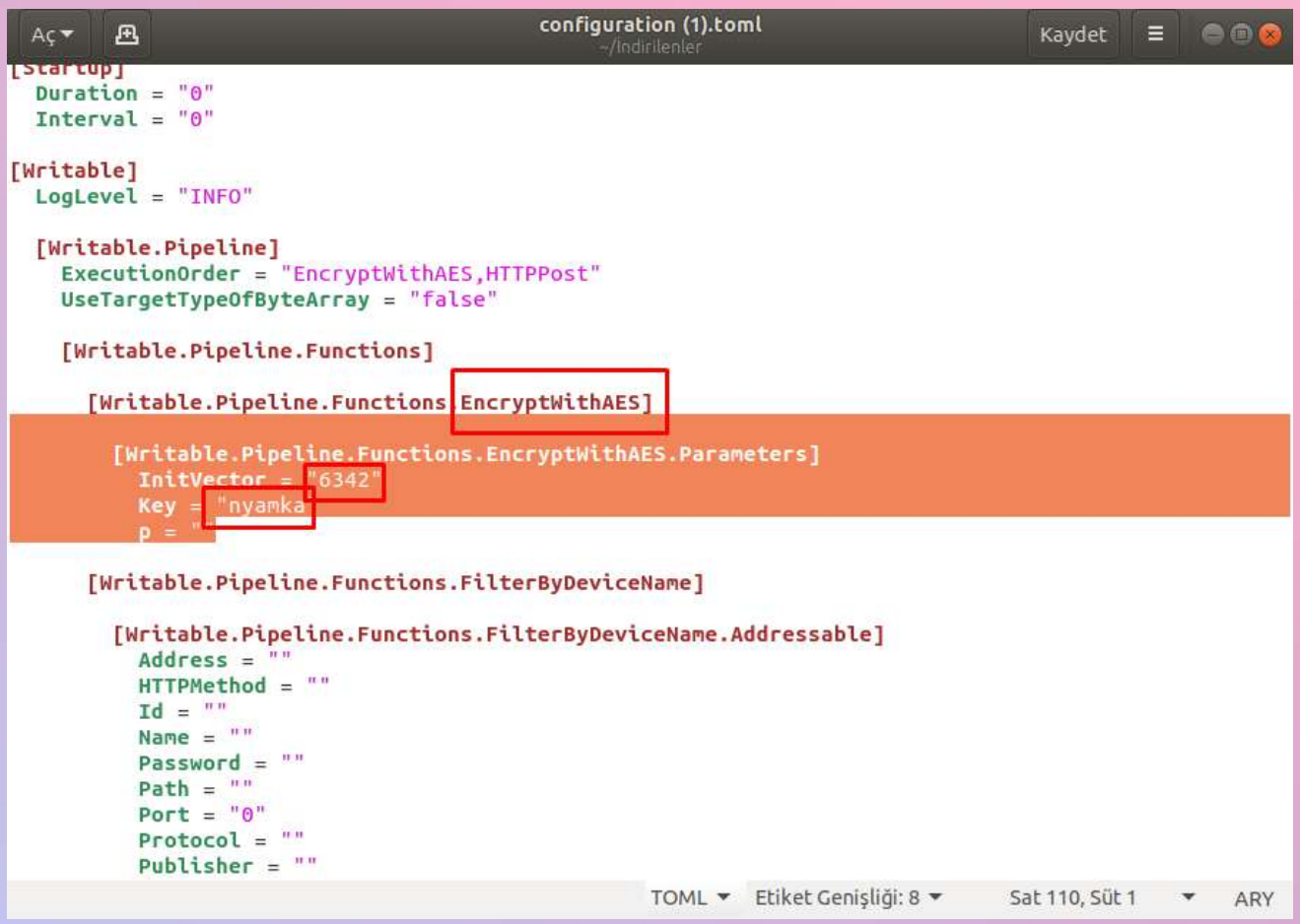
By following the steps I will show on the Consul interface, let's view the "Key" and "InitVector" values entered as parameters. For "Key";

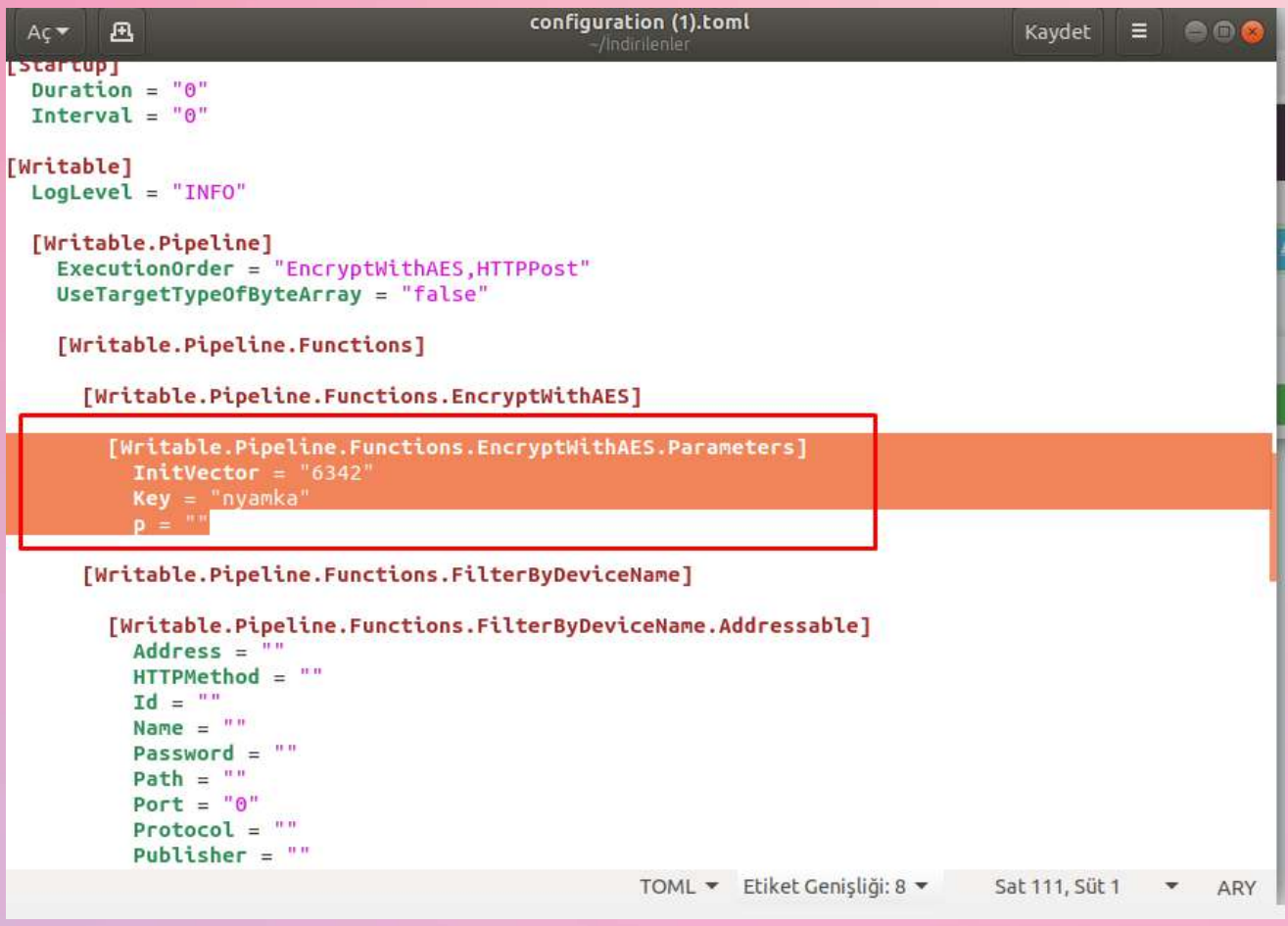


for "InitVector" ;

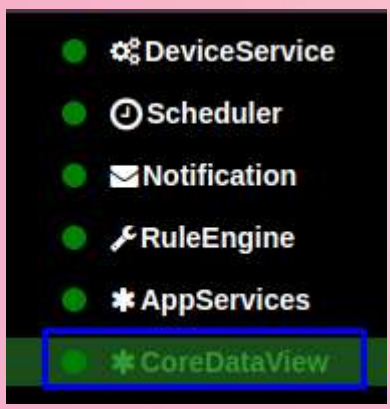


All other services and entered parameters can be viewed in this way. If you want to download and review the current configuration file of the selected application service, you can click the "Download Configuration" button. The downloaded ".toml" file contains some information, including the added services and values entered as parameters.

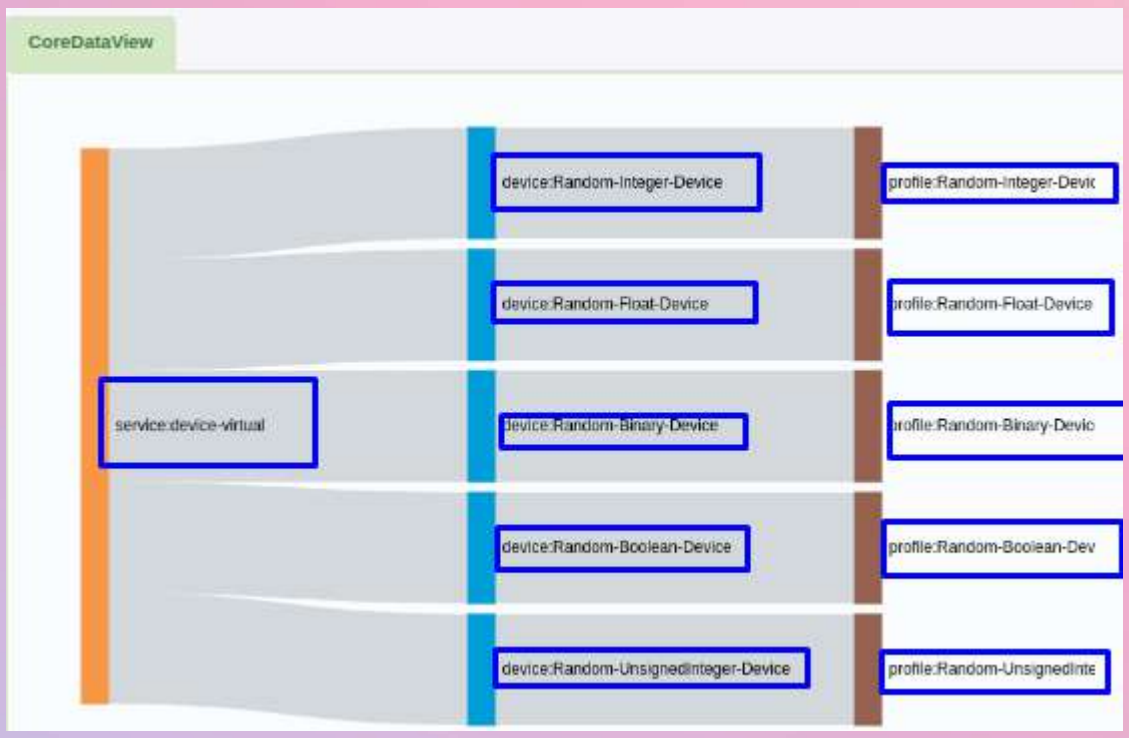


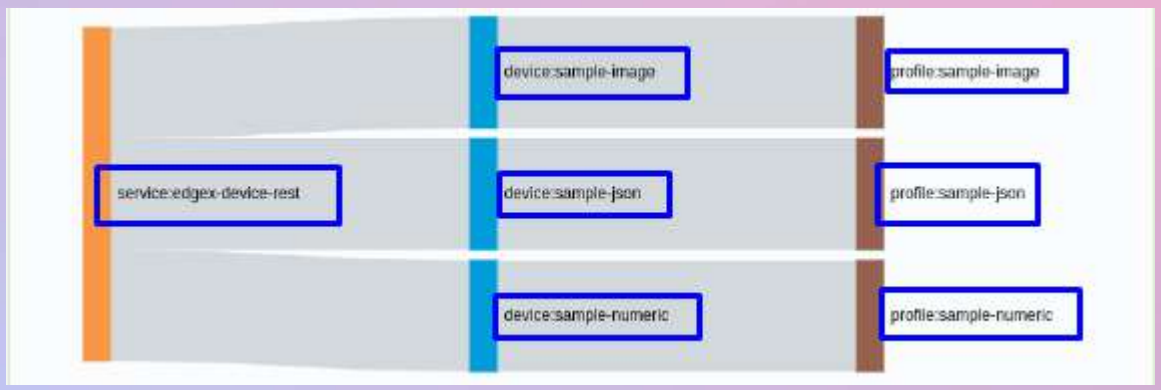

Aç
configuration (1).toml
~/İndirilenler
Kaydet
[Startup]
Duration = "0"
Interval = "0"
[Writable]
LogLevel = "INFO"
[Writable.Pipeline]
ExecutionOrder = "EncryptWithAES,HTTPPost"
UseTargetTypeOfByteArray = "false"
[Writable.Pipeline.Functions]
[Writable.Pipeline.Functions.EncryptWithAES]
[Writable.Pipeline.Functions.EncryptWithAES.Parameters]
InitVector = "6342"
Key = "nyamka"
p = ""
[Writable.Pipeline.Functions.FilterByDeviceName]
[Writable.Pipeline.Functions.FilterByDeviceName.Addressable]
Address = ""
HTTPMethod = ""
Id = ""
Name = ""
Password = ""
Path = ""
Port = "0"
Protocol = ""
Publisher = ""
TOML
Etiket Genişliği: 8
Sat 111, Süt 1
ARY

COREDATAVIEW



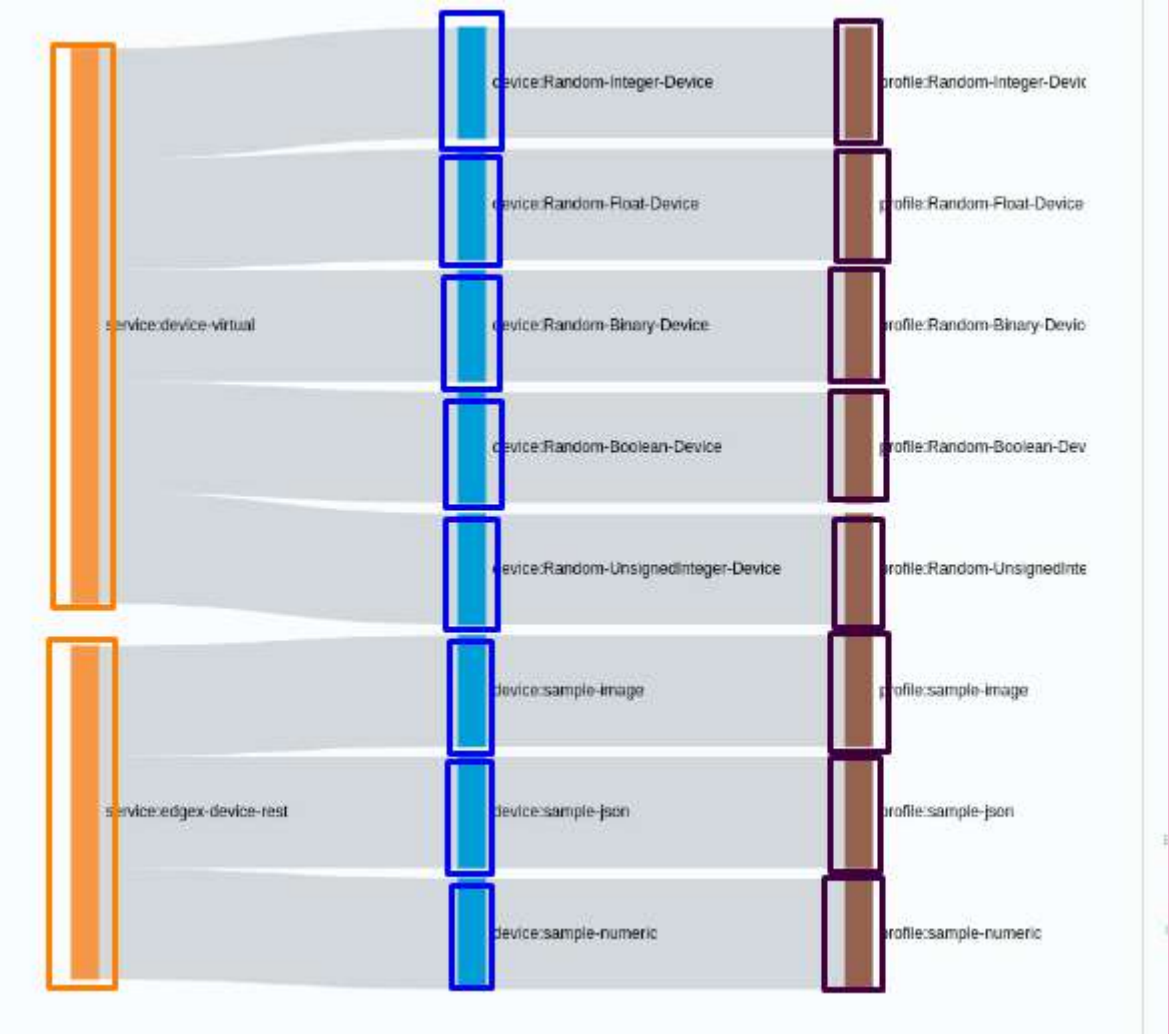
This tab is where inspection and monitoring operations are performed. Virtual devices and REST devices are displayed separately in this window.





When clicked, the rectangular information boxes below are as follows, according to the text in the right-side tab:

Orange rectangles: provide device service information.
Blue rectangles: provide device information.
Brown rectangles: provide device profile information.



"Service: edgex-device-rest" when clicked on the orange box on the left side of the table, "DeviceService" information is obtained.

Device Service

#	ID	Name	Description	Labels	Addressable	OperatingState	AdminState	Devices	Created Time	Modified Time
1	2d34d4a5-0033-47bd-ba71-501f0df9440a	edgex-device-rest				ENABLED	UNLOCKED		2021-01-05 00:11:23	2021-01-05 00:11:23

The "Addressable" and "Devices" buttons on this table are active and information is given in a new table below.

Device Service

#	ID	Name	Description	Labels	Addressable	OperatingState	AdminState	Devices	Created Time	Modified Time
1	2d34d4a5-0033-47bd-ba71-501f0df9440a	edgex-device-rest				ENABLED	UNLOCKED		2021-01-05 00:11:23	2021-01-05 00:11:23

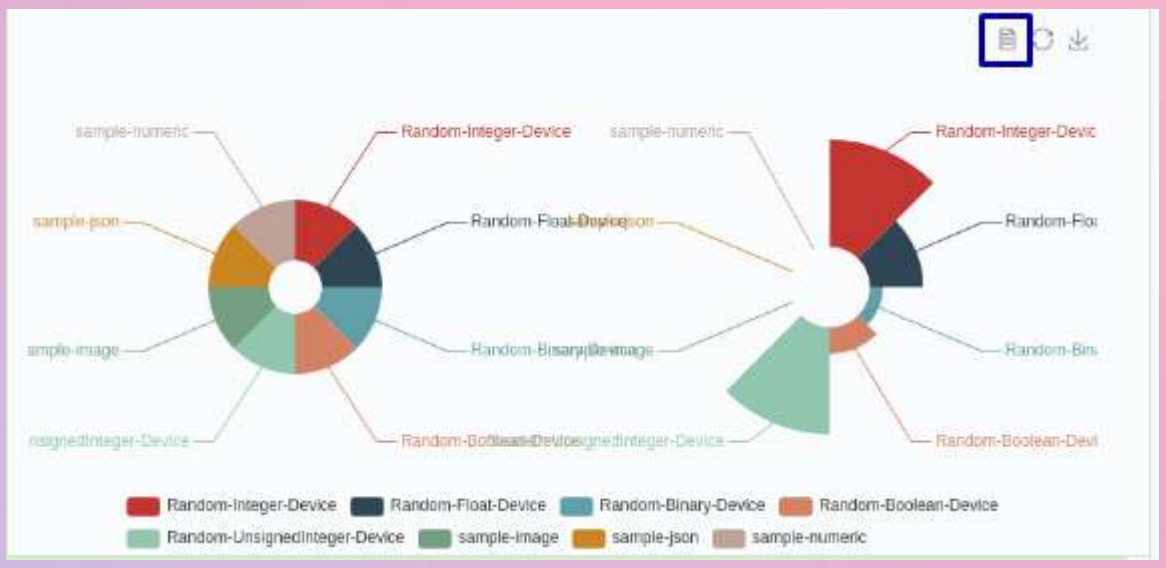
Addressable

ID	Name	Protocol	Address	Port	Path	Created Time	Modified Time
69c2f2ed-1d5a-4064-a215-3f60b55bf483	edgex-device-rest	HTTP	edgex-device-rest	49986	/api/v1/callback	2021-01-05 00:11:23	2021-01-05 00:11:23

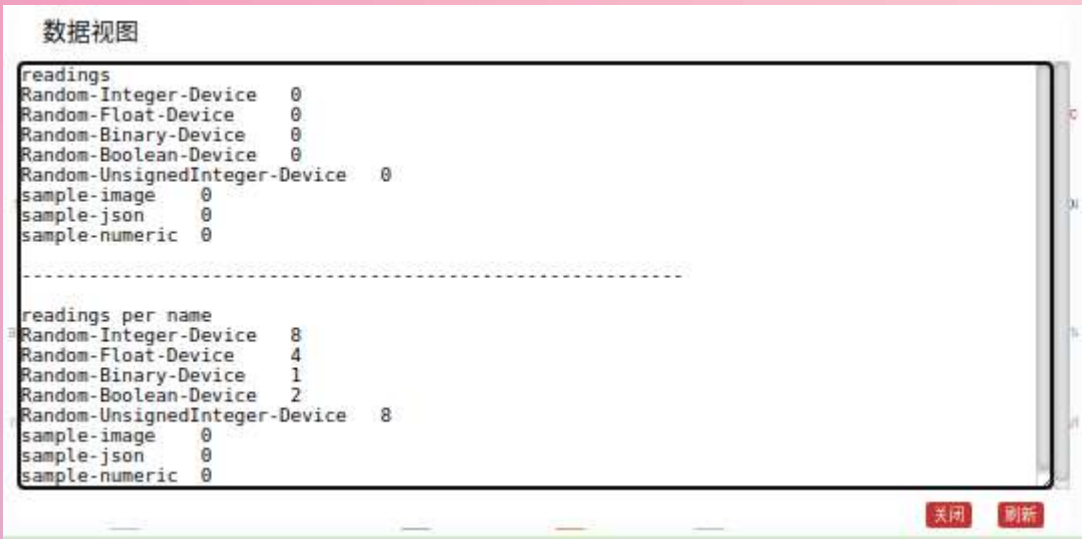
Devices

#	ID	Name	Description	Labels	Profile	OperatingState	AdminState	Events	Readings	Created Time	Modified Time
1	ae13cc14-76e6-4db9-952e-9ab25afd5e87	sample-image	RESTful Device that sends in binary image data	rest,binary,image	sample-image	ENABLED	UNLOCKED			2021-01-05 00:11:23	2021-01-05 00:11:23
2	2352c6b2-de91-4818-ba42-9a1d634d9fc4	sample-json	RESTful Device that sends in JSON data	rest,json	sample-json	ENABLED	UNLOCKED			2021-01-05 00:11:23	2021-01-05 00:11:23
3	d47b5112-c32a-4aee-aadd-35abe9be1c2d	sample-numeric	RESTful Device that sends in numeric data	rest,numeric,float,int	sample-numeric	ENABLED	UNLOCKED			2021-01-05 00:11:23	2021-01-05 00:11:23

The left circle graph on the bottom right side of the page shows the status of data perception from a device or sensor. The right circle graph shows the number of values that can be entered based on the device name. When we hover over the slice of the circle, it displays information as a percentage.



When clicking on the symbol marked on the top right of the graphs, it outputs a reading in text form.



On the upper right side of the page, there is a "event-reading" graph. For a value to be present in this graph, a read value must first exist. The option of "event" alone would be meaningless. A read value must exist so that an event occurs. In our example, no value will be seen here as there is no real device or sensor.

ADD DEVICE

We have reached the end of using the "Golang" interface, which is the interface of EdgeX Foundry. Now, we will add an IoT device. The device we will add will be a Raspberry Pi 3b+. We will show how to send humidity (humidity) and temperature (temperature) values to EdgeX and how to integrate this device into EdgeX. We have said that there are many interfaces for EdgeX. We will use one of them, "Postman". We had installed Postman to test if EdgeX was working when we installed EdgeX.

We have also shown that we can open Postman from the interface of Ubuntu. If you want to open it from the terminal screen, we download "screen" through the apt package installer.

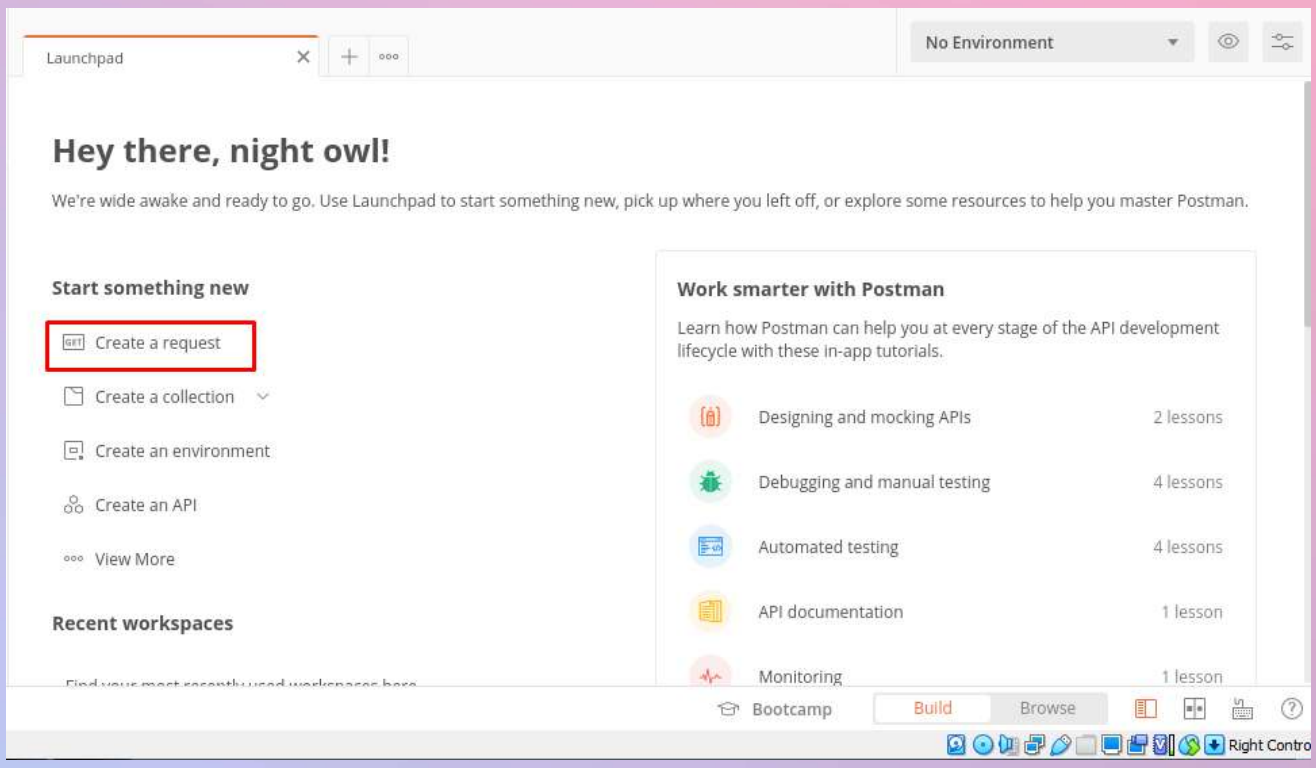
sudo apt install screen

```
edgexfoundry@edgexfoundry-VirtualBox:~$ sudo apt install screen
[sudo] password for edgexfoundry:
Paket listeleri okunuyor... Bitti
Bağımlılık ağacı oluşturuluyor
Durum bilgisi okunuyor... Bitti
Aşağıdaki paketler otomatik olarak kurulmuş ve artık bu paketlere gerek duyulmuy
or:
  linux-headers-5.4.0-58-generic linux-hwe-5.4-headers-5.4.0-42
  linux-hwe-5.4-headers-5.4.0-58 linux-image-5.4.0-58-generic
  linux-modules-5.4.0-58-generic linux-modules-extra-5.4.0-58-generic
Bu paketleri kaldırmak için 'sudo apt autoremove' komutunu kullanın.
Aşağıdaki ek paketler kurulacak:
  libutempter0
Önerilen paketler:
  byobu | screenie | iselect ncurses-term
Aşağıdaki YENİ paketler kurulacak:
  libutempter0 screen
0 paket yükseltilecek, 2 yeni paket kurulacak, 0 paket kaldırılacak ve 12 paket
yükseltilmeyecek.
572 kB arşiv dosyası indirilecek.
Bu işlem tamamlandıktan sonra 1.052 kB ek disk alanı kullanılacak.
N: '/etc/apt/sources.list.d/' dizinindeki 'google-chrome.list.save' dosyası geçe
rsiz bir dosya uzantısı olduğu için yok sayılıyor
Devam etmek istiyor musunuz? [E/h] e
```

After installing "screen", we can start Postman using the command in the terminal screen.

screen postman

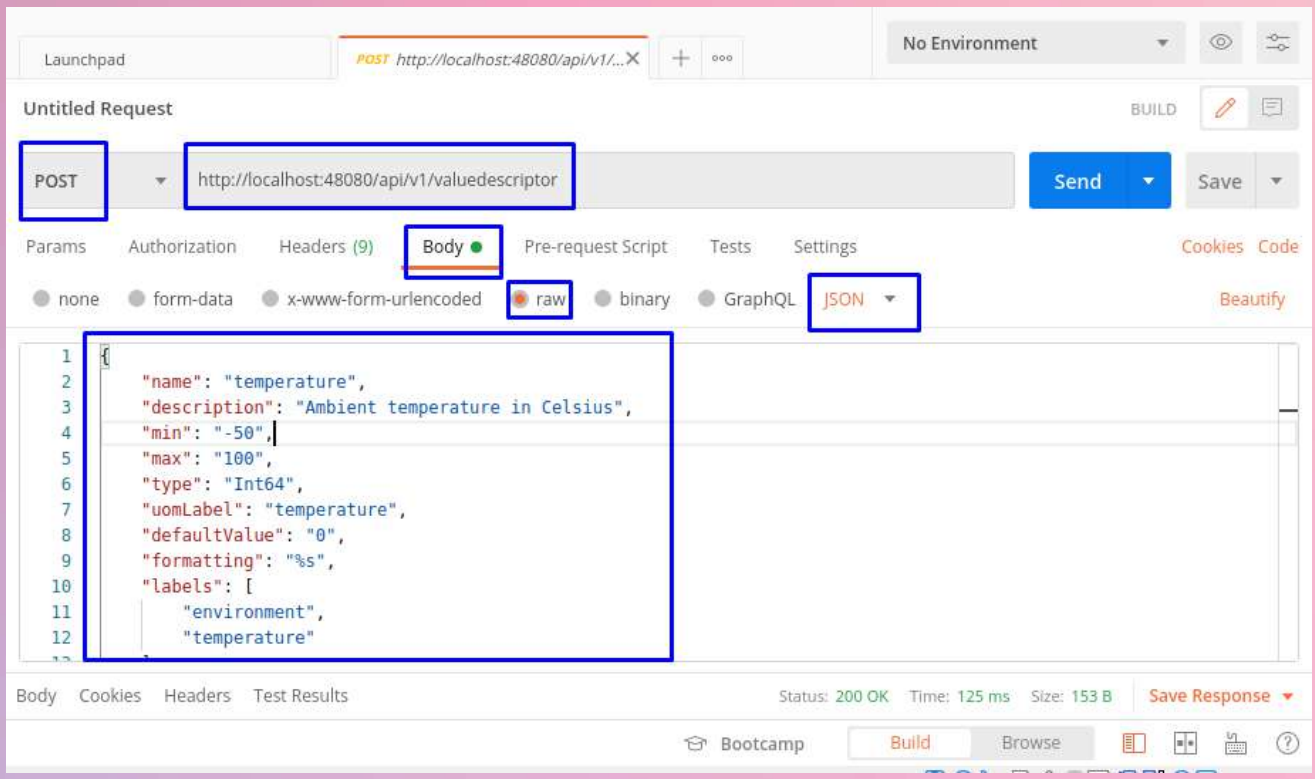
Once Postman is opened, we proceed with our process by clicking on the "Create a request" tab.



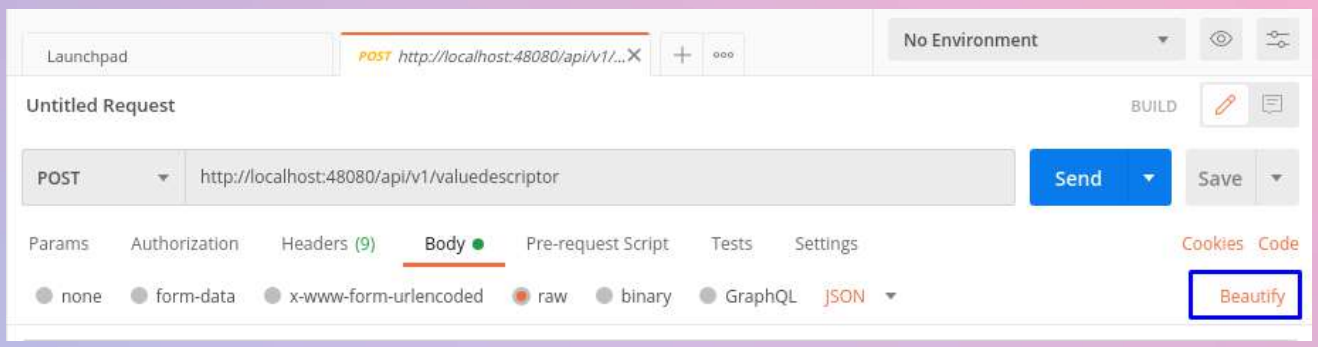
We define the temperature value.

Method : POST
url : http://localhost:48080/api/v1/valuedescriptor
Body : raw and JSON

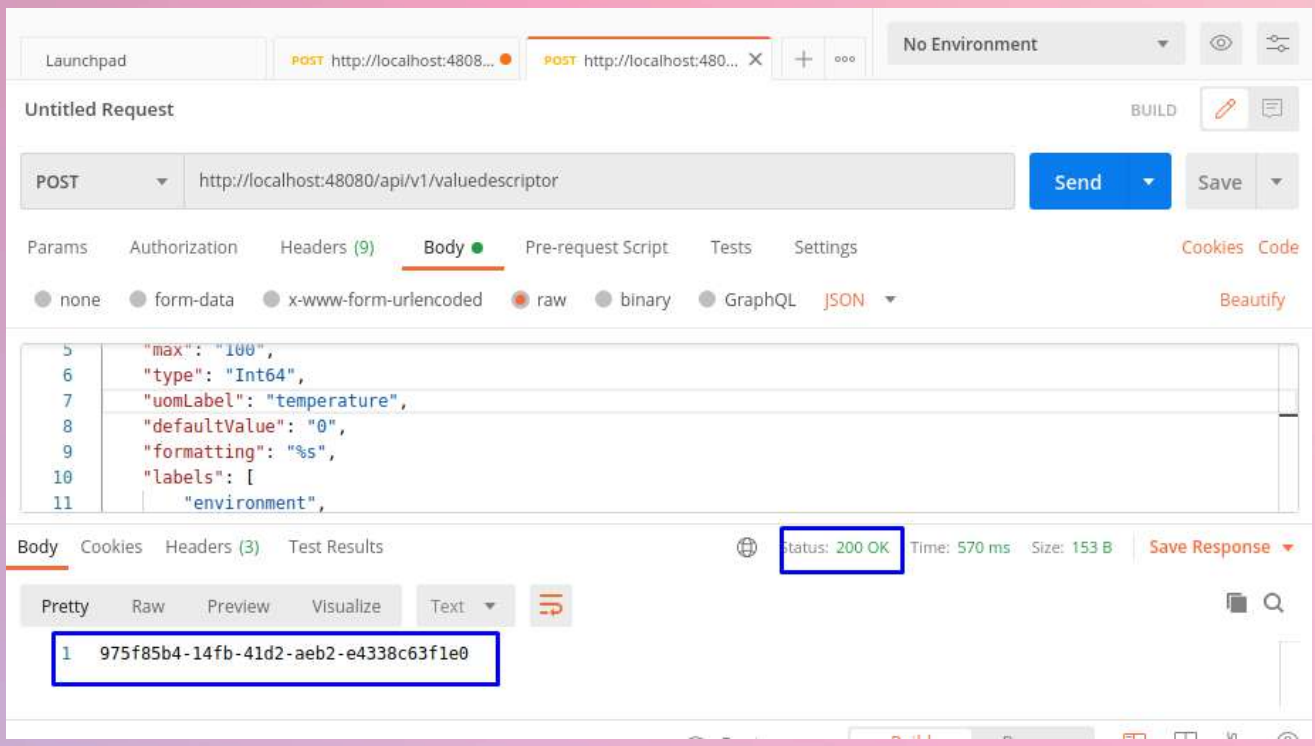

```
{
    "name": "temperature",
    "description": "Ambient temperature in Celsius",
    "min": "-50",
    "max": "100",
    "type": "Int64",
    "uomLabel": "temperature",
    "defaultValue": "0",
    "formatting": "%s",
    "labels": [
        "environment",
        "temperature"
    ]
}
```



If an irregular image occurs when you copy and paste while defining the value, you can automatically arrange it with the "Beautify" button.



We complete the temperature value definition process with the "Send" button. If we get a "200 OK" output, the process has been successfully completed. As can be seen in the screen shot, a unique identity is generated for us (there is no need to note these identities that will be generated).

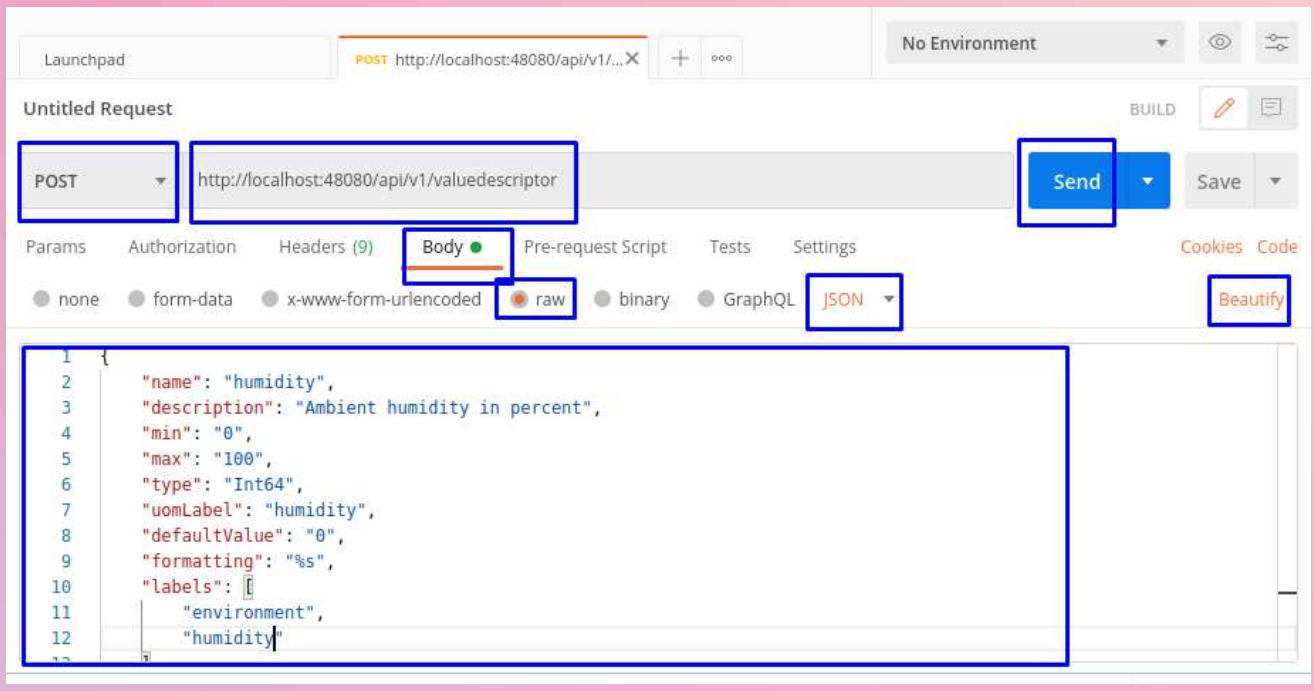


We define the humidity value.

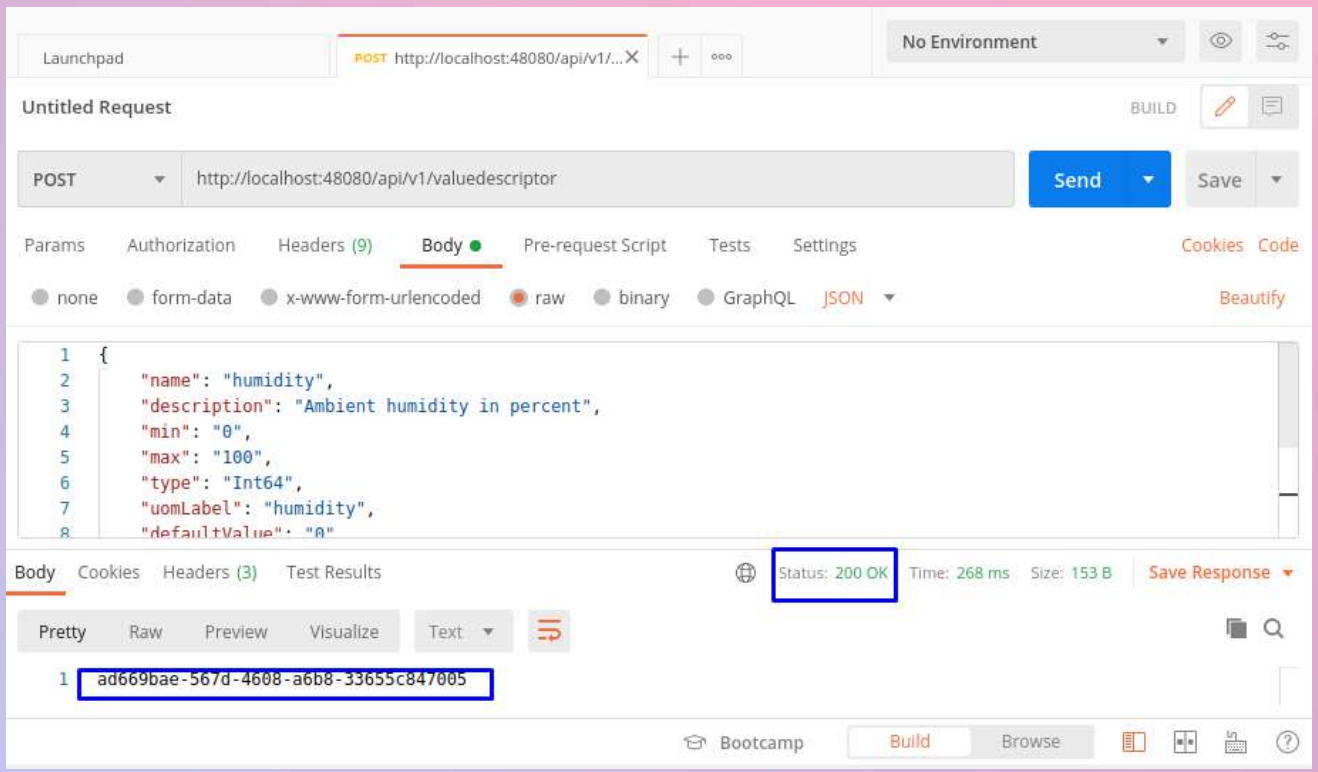
Method : POST
url : http://localhost:48080/api/v1/valuedescriptor
Body : raw and JSON

```
{
   "name": "humidity",
   "description": "Ambient humidity in percent",
   "min": "0",
   "max": "100",
   "type": "Int64",
   "uomLabel": "humidity",
   "defaultValue": "0",
   "formatting": "%s",
   "labels": [
      "environment",
      "humidity"
```

```
    ]
}
```



If we have successfully identified the humidity value, a unique identity will still be created.

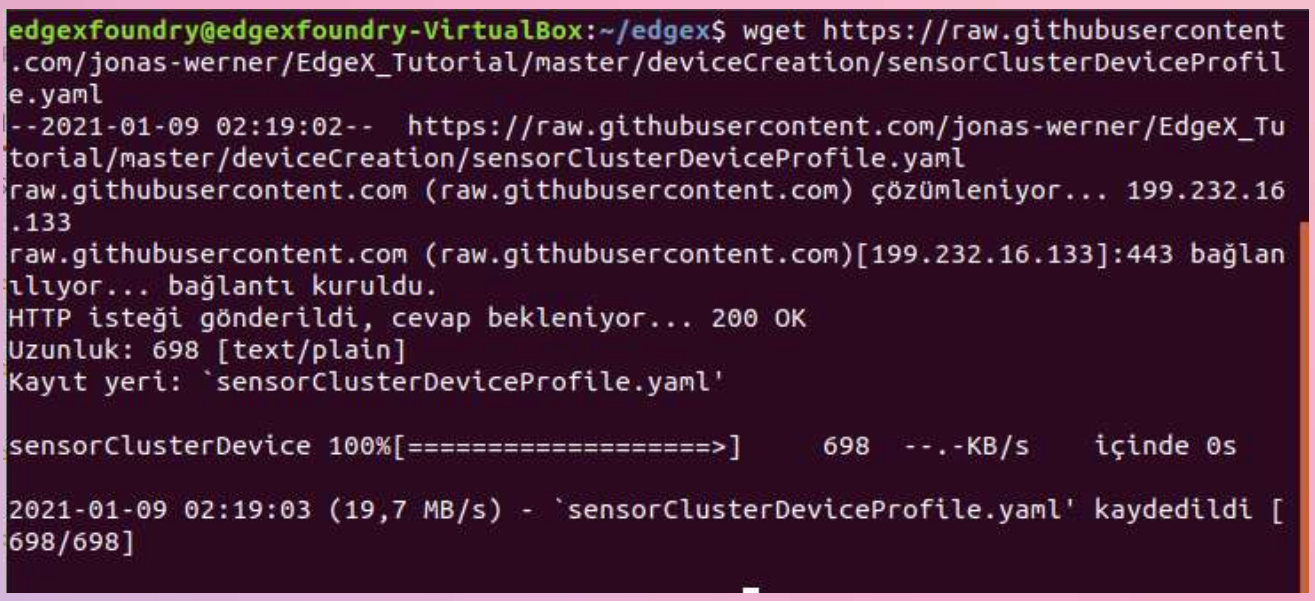


After defining the temperature and humidity values, we need to add the device profile.This device profile ;

https://raw.githubusercontent.com/jonas-werner/EdgeX_Tutorial/master/deviceCreation/sensorClusterDeviceProfile.yaml

You can download it from the mentioned address. You can either do this by using the terminal screen or your web browser's "Save As" option. To download from the terminal screen:

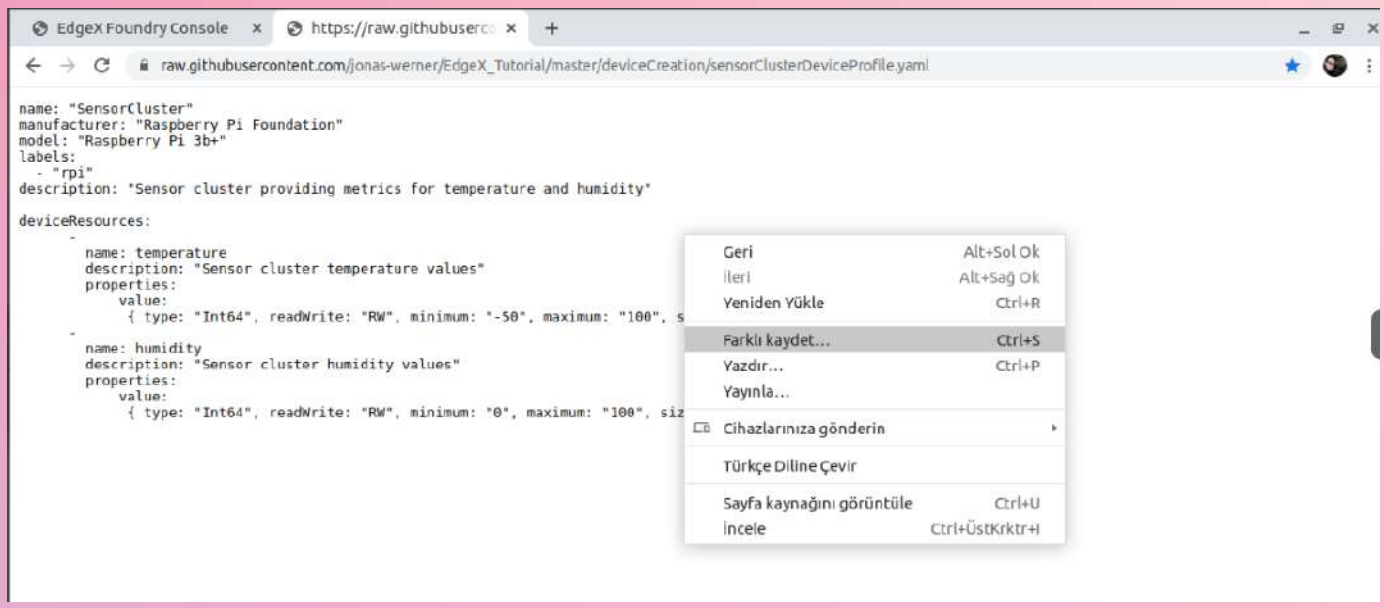
wget https://raw.githubusercontent.com/jonas-werner/EdgeX_Tutorial/master/deviceCreation/sensorClusterDeviceProfile.yaml

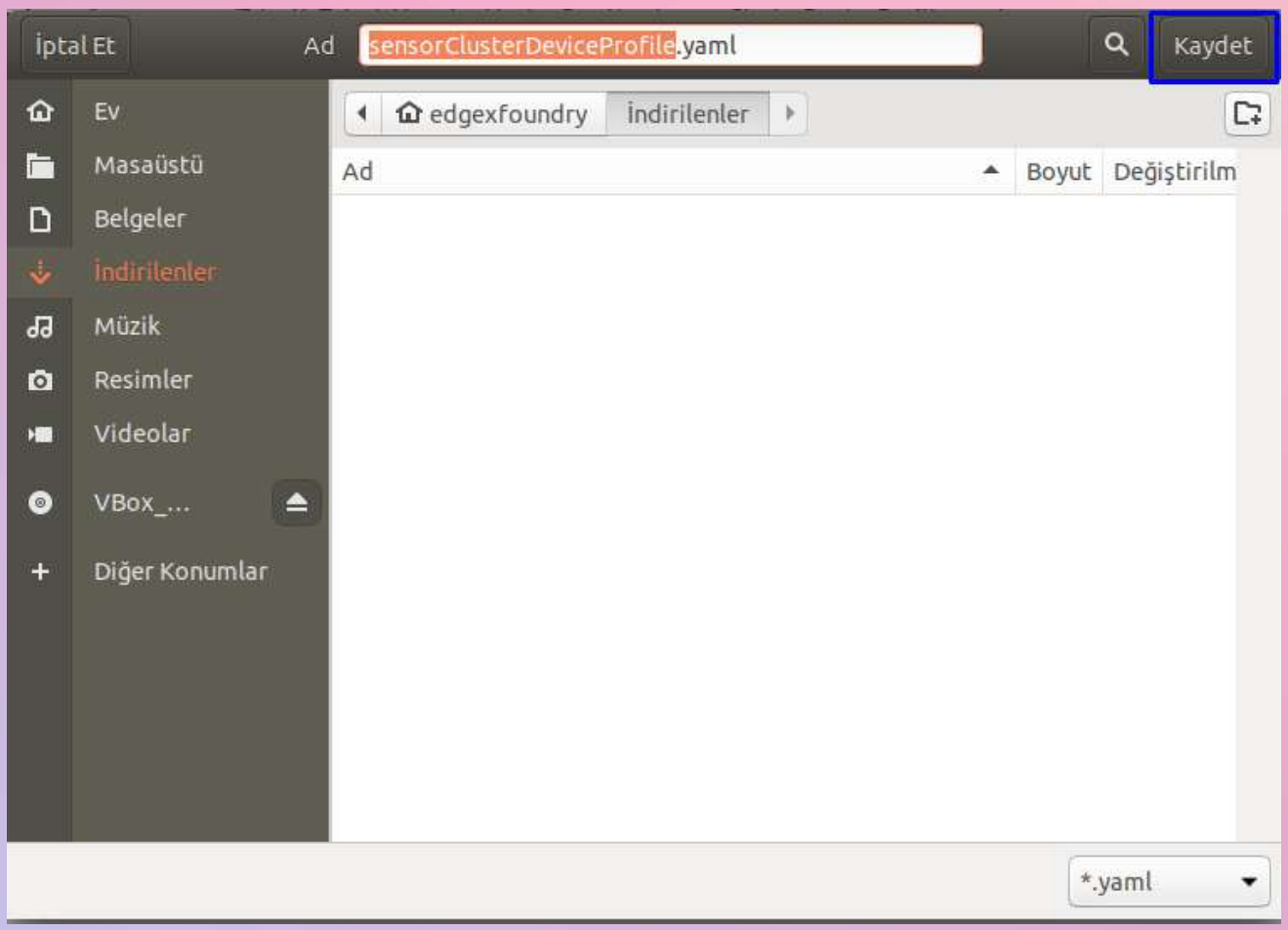


To download from the web browser:

https://raw.githubusercontent.com/jonas-werner/EdgeX_Tutorial/master/deviceCreation/sensorClusterDeviceProfile.yaml

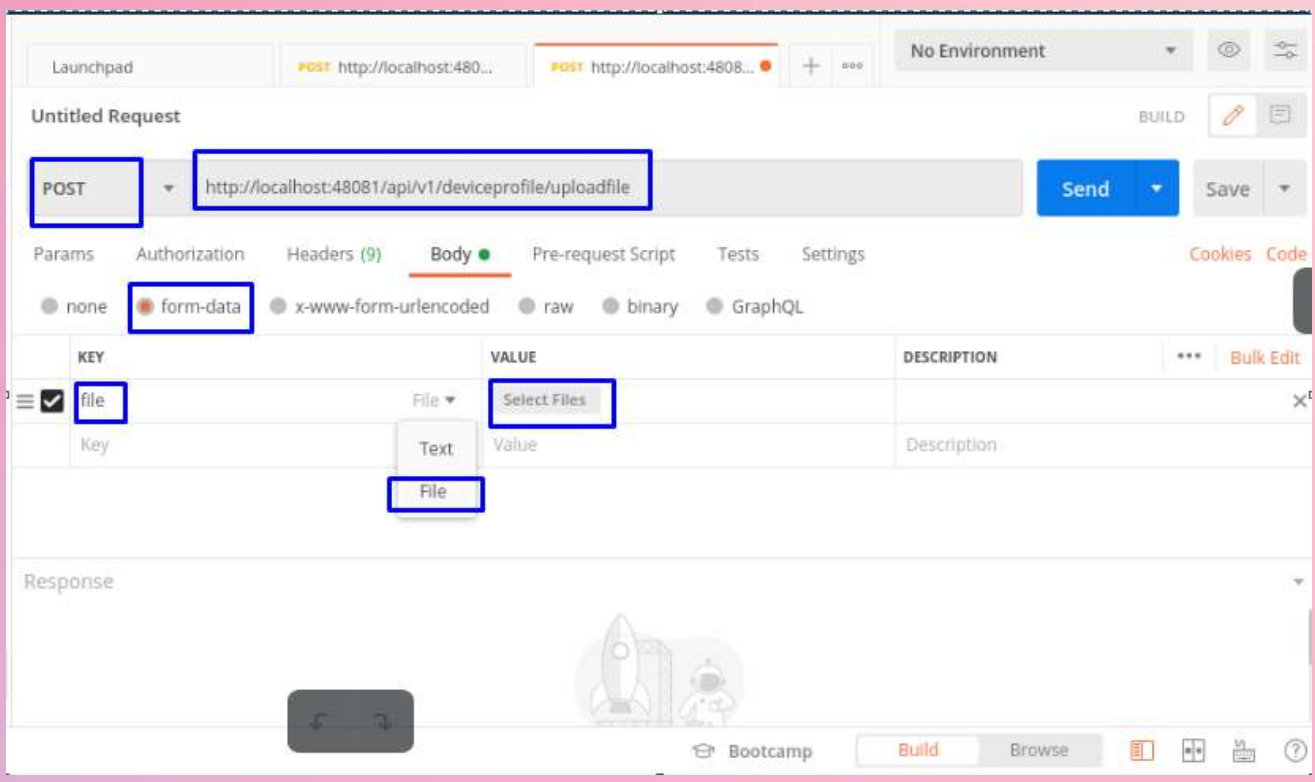
Go to the address, right-click on the page and click "Save As", then select the directory where the file will be saved and click the "Save" button to download the file.



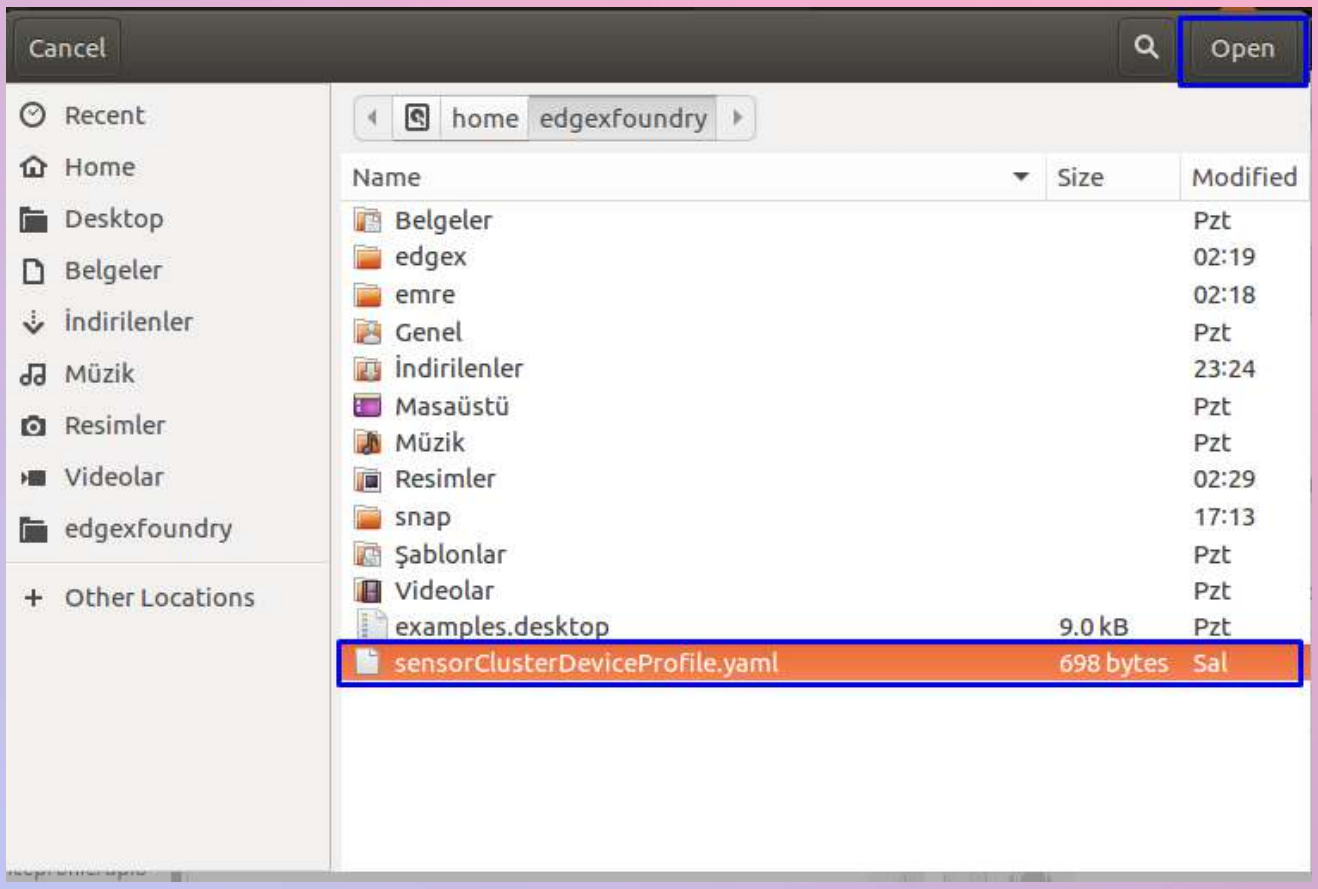


To add this device profile in Postman:

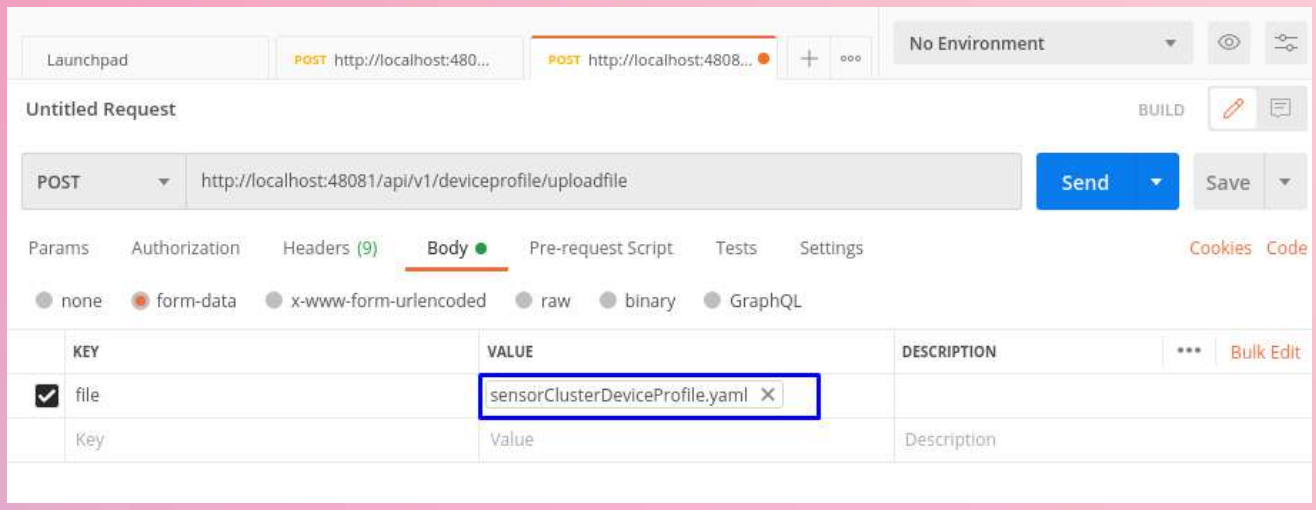
Method : POST
url : http://localhost:48081/api/v1/deviceprofile/uploadfile
Body : form-data
Hover over the Key and change it to "File".



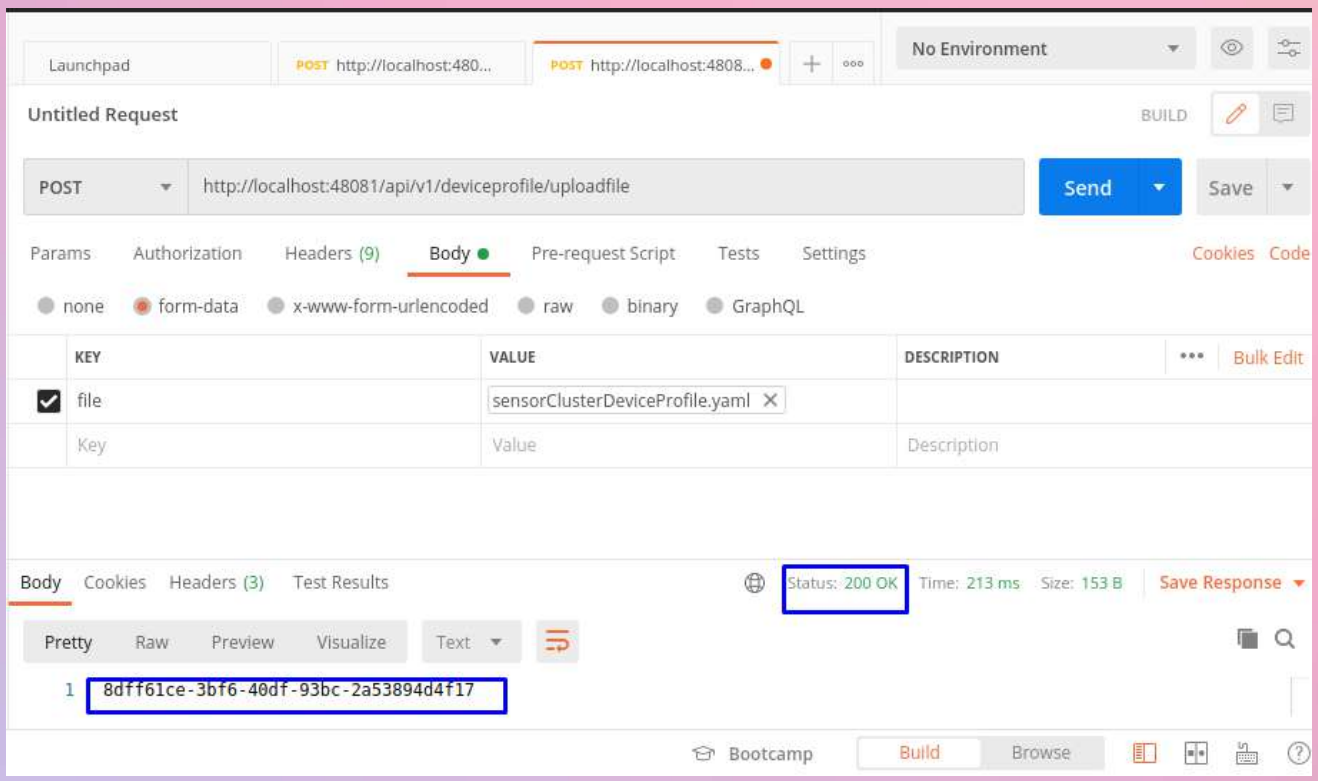
By hovering over the Key and changing it to "File" and writing "file" in the key section, the "Select Files" option will appear on the right. Click on this option, select the "sensorClusterDeviceProfile.yaml" file that you recently downloaded, and click "Open".



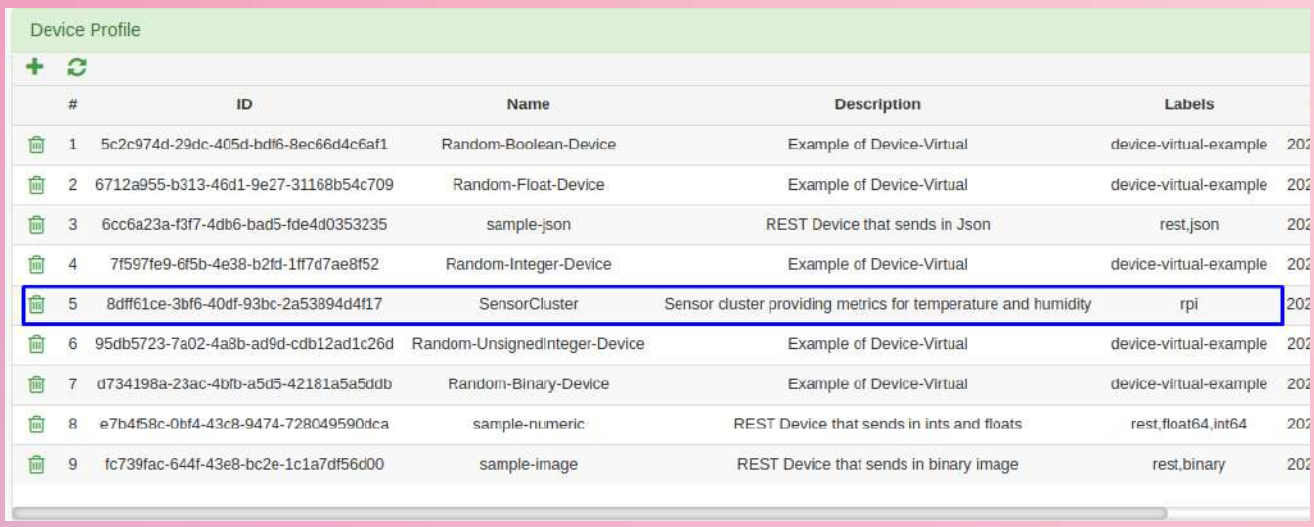
The device profile file name will appear after you have selected the device profile file.



After clicking the "Send" button, a unique identity will be created again. Some companies provide the device profile file to the user.



Once you have successfully created this device profile, the device profile will be created in the EdgeX interface.

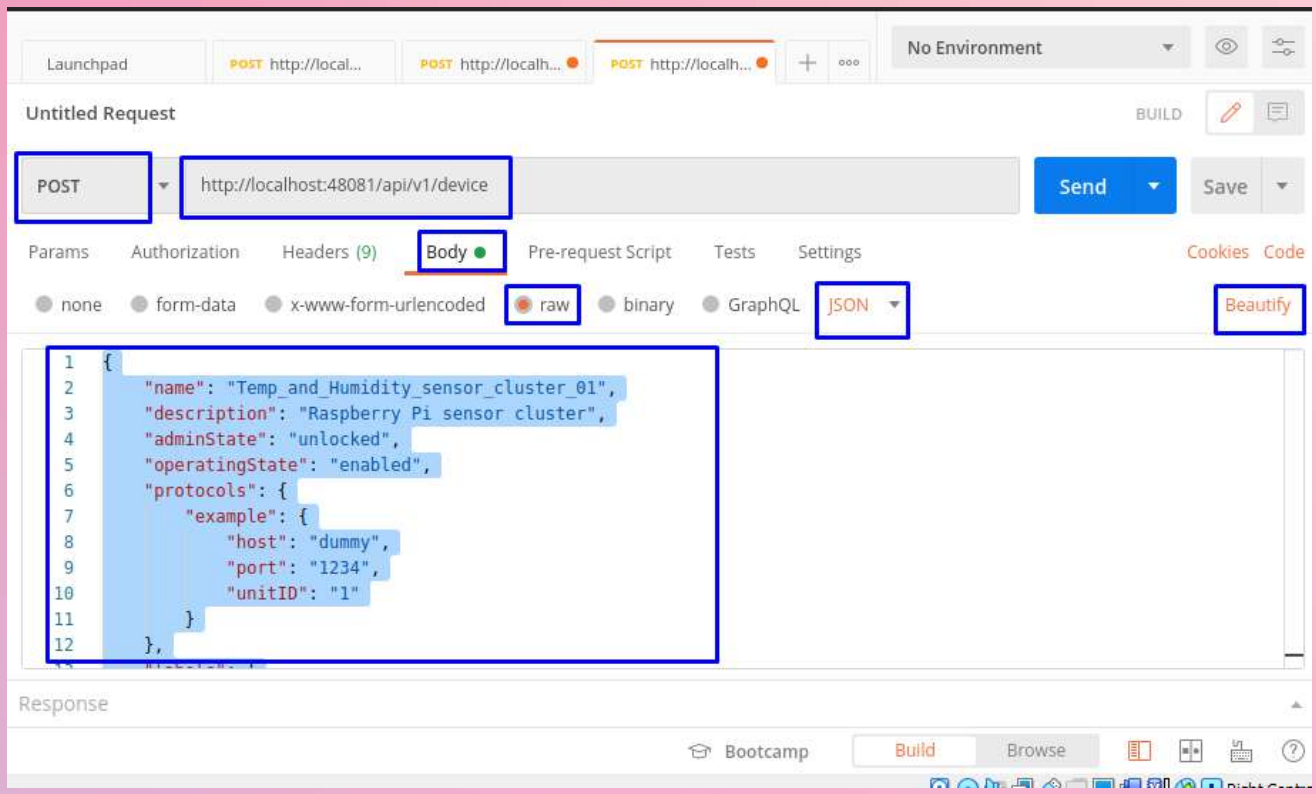


After creating the device profile, you need to create the device.

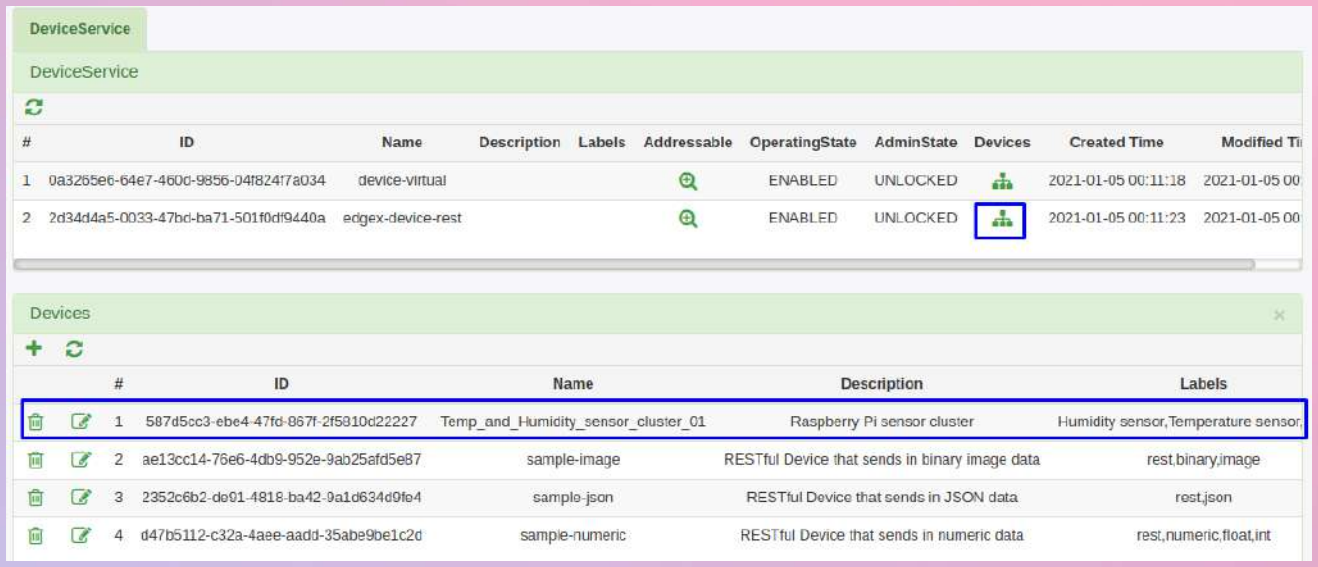
Method : POST
url : http://localhost:48081/api/v1/device
Body : raw and JSON

```
{
"name": "Temp_and_Humidity_sensor_cluster_01",
"description": "Raspberry Pi sensor cluster",
"adminState": "unlocked",
"operatingState": "enabled",
"protocols": {
"example": {
"host": "dummy",
"port": "1234",
"unitID": "1"
}
},
"labels": [
"Humidity sensor",
"Temperature sensor",
"DHT11"
],
"location": "Tokyo",
"service": {
"name": "edgex-device-rest"
},
"profile": {
"name": "SensorCluster"
}
}
```

You can edit it using the "Beautify" button.

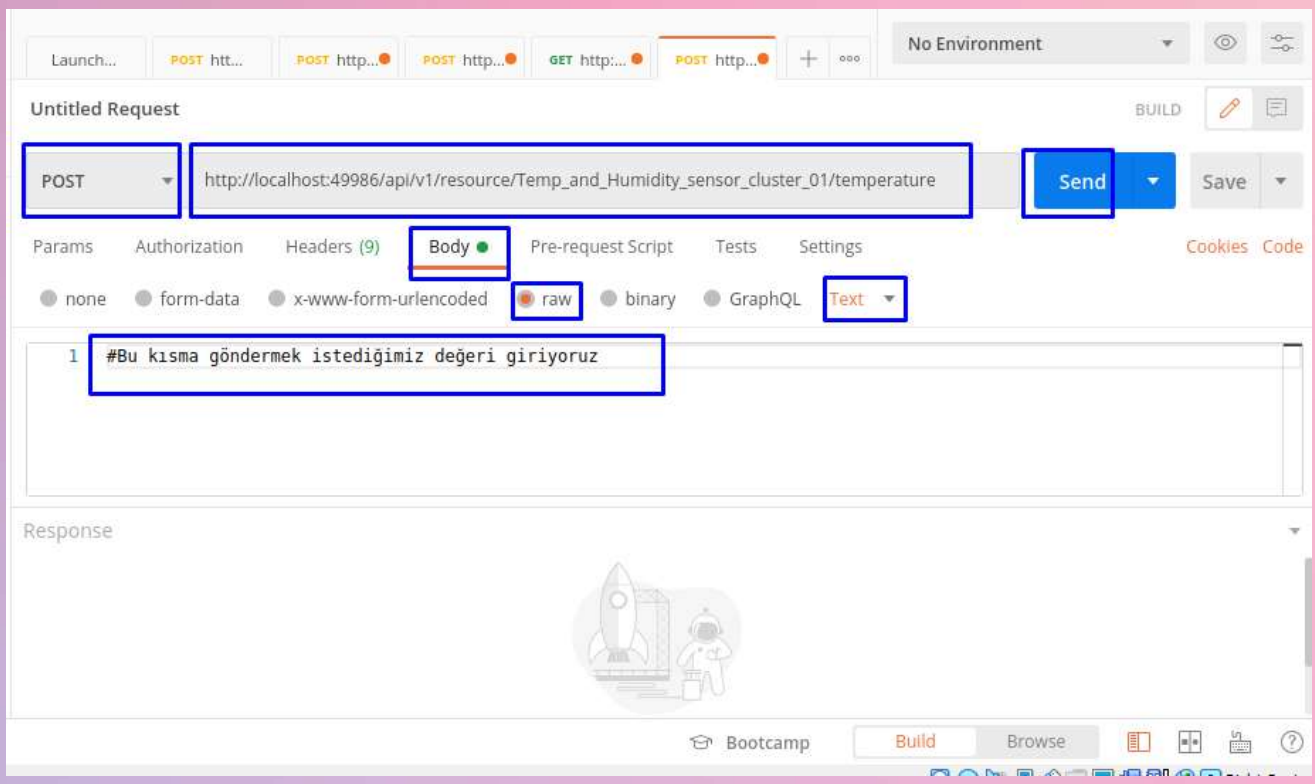


After sending it with "Send", there will be a new REST device in the EdgeX interface.

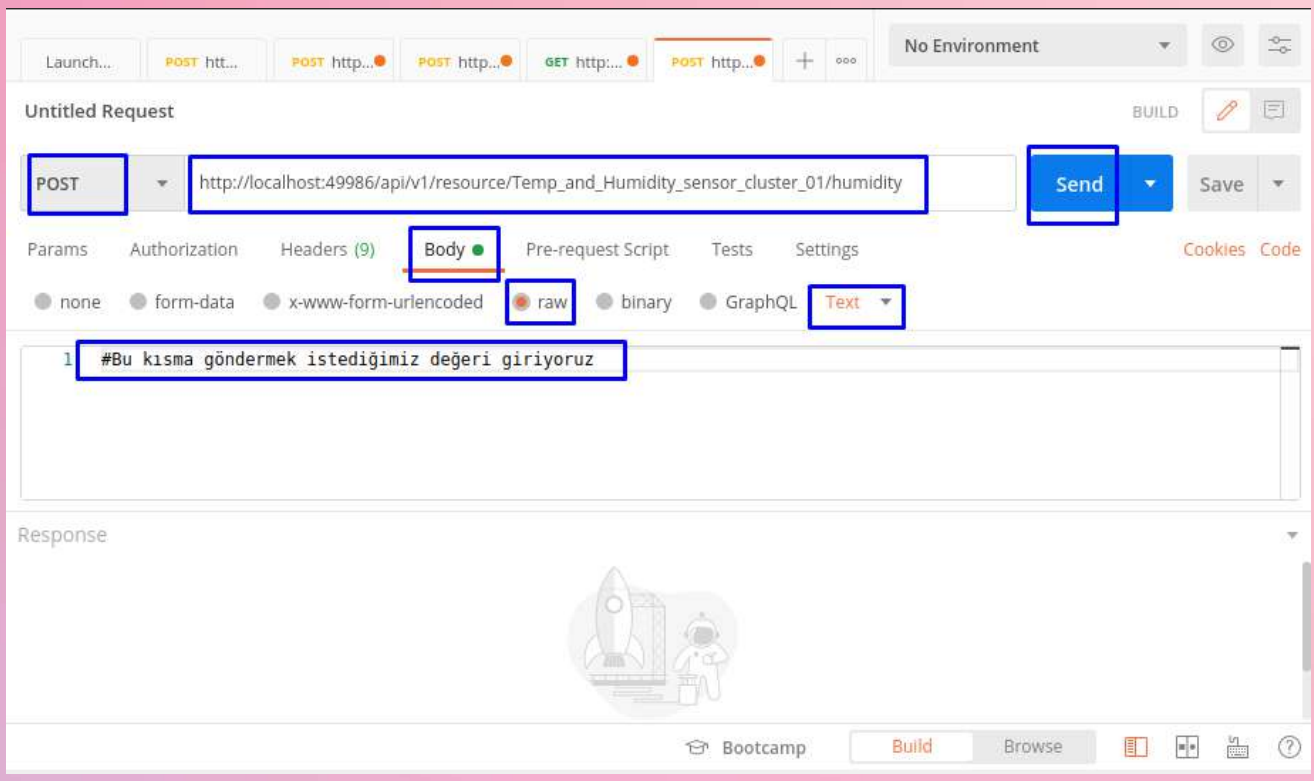


Now we can send the humidity and temperature values to the device that we added to EdgeX.

Method: POST
url : http://localhost:49986/api/v1/resource/Temp_and_Humidity_sensor_cluster_01/temperature
Body : raw and text



The "temperature" at the end of the URL is for sending the temperature value. To send the humidity value, we can change this to "humidity".

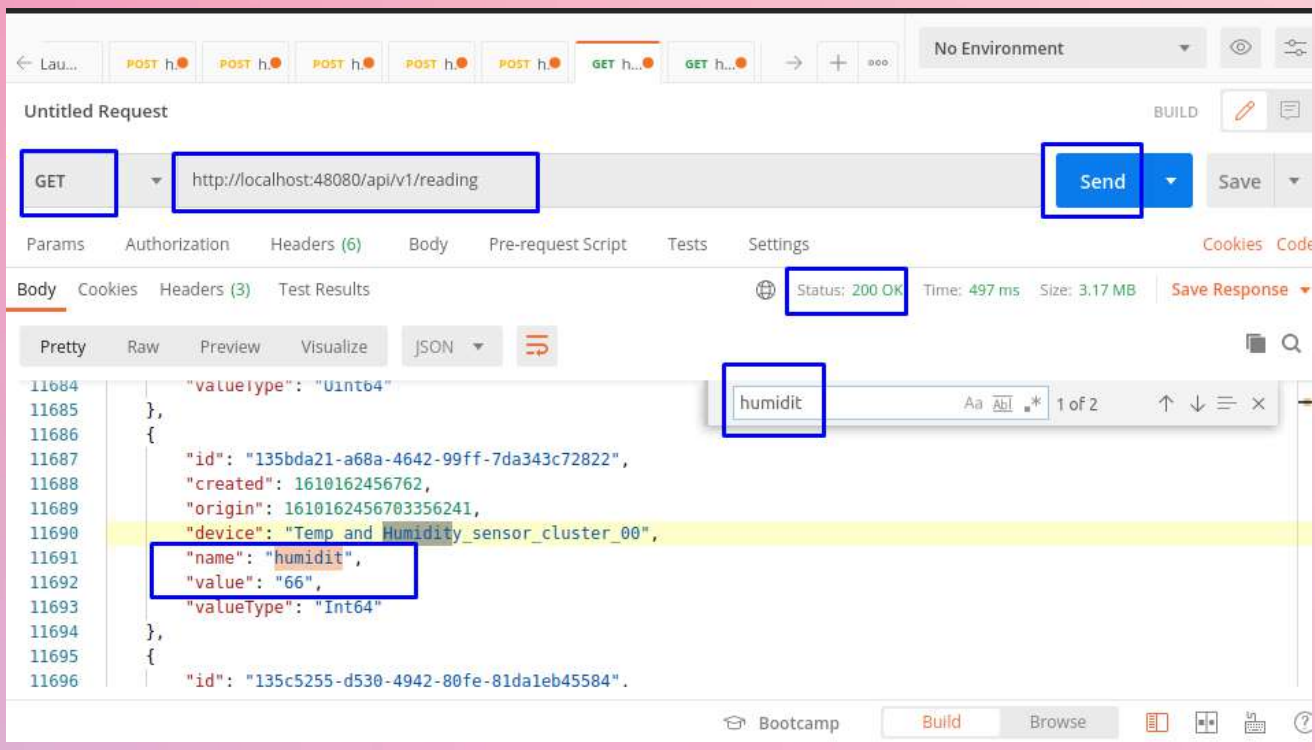


For example, let's send the number 66 for the humidity value.

To view the value that we sent and the values that have been previously sent and saved in the database:

Method : GET
url : http://localhost:48080/api/v1/reading

This request returns a JSON result by default.

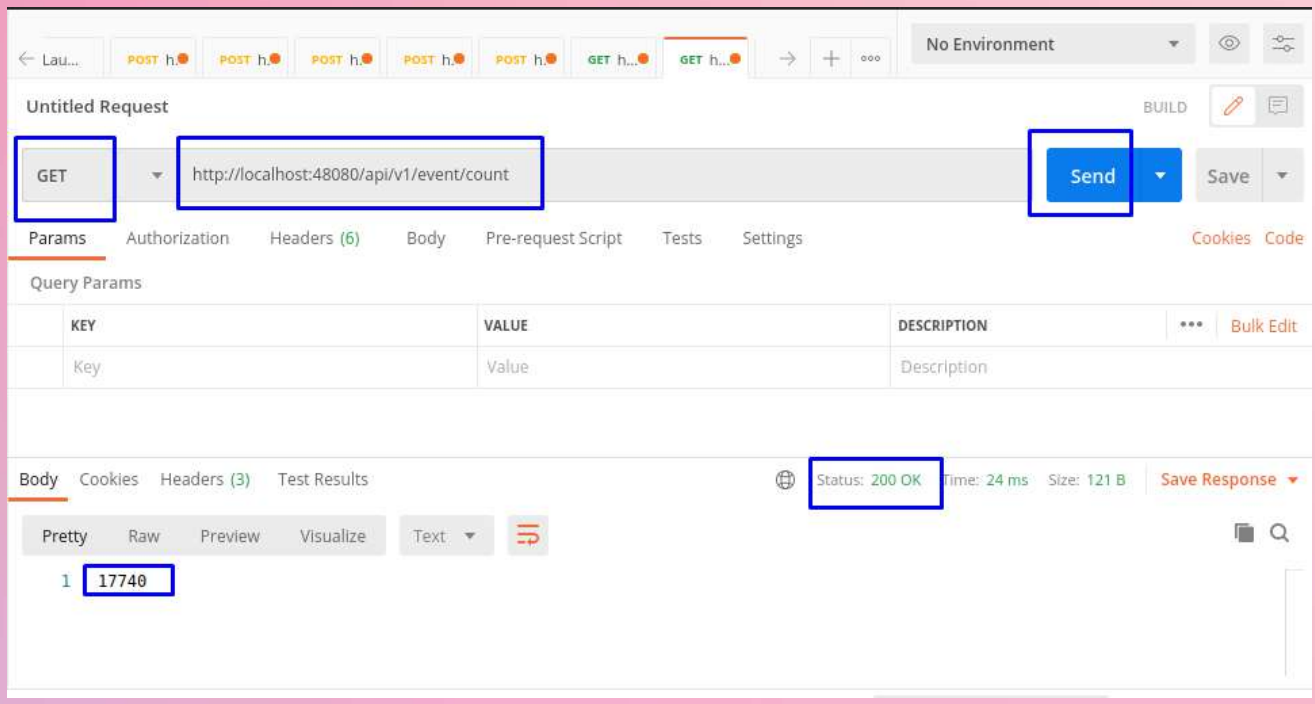


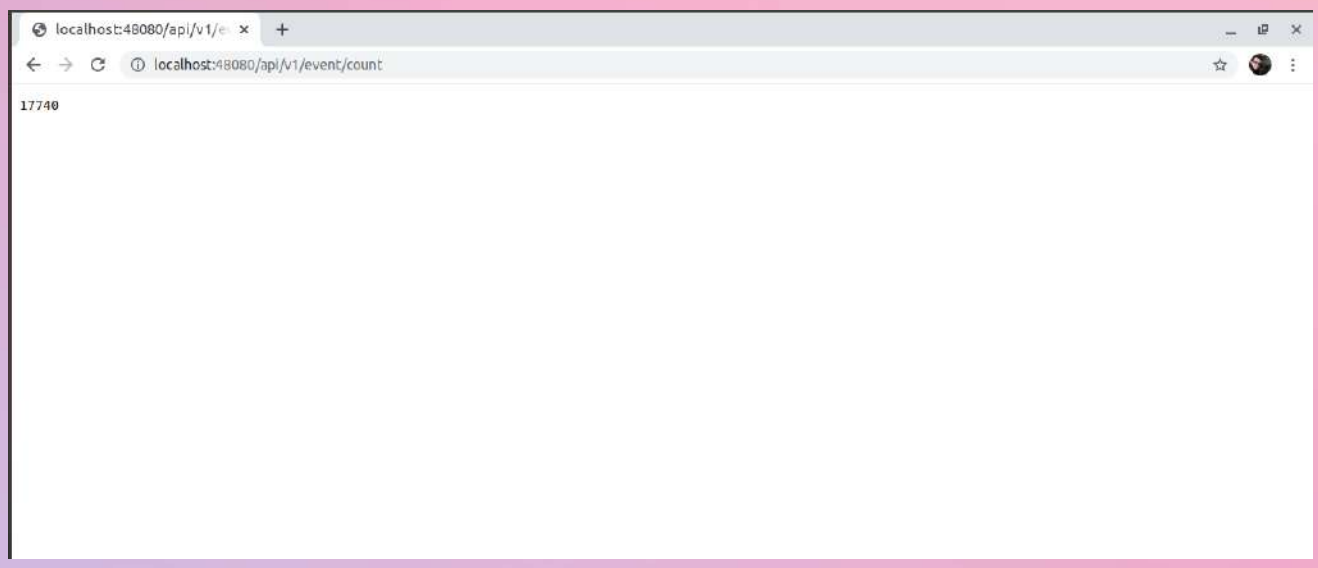
As seen, there are a large number of data stored in the database. This includes the value 66 that we sent. Since there is a large amount of data, we can search for the data we want to see using the Ctrl+F key combination.

Finally, we can view the number of events that have taken place. For this;

Method : GET
url : http://localhost:48080/api/v1/event/count

If you are querying the number of events through Postman, you can view the number of events once. New events do not change on this screen, and you need to make a request again using the same method and URL. If you search the same URL from your browser, the result will be the same as in Postman and the event count will be updated every time you refresh your browser with the F5 key.


No Environment
Untitled Request
BUILD
GET
http://localhost:48080/api/v1/event/count
Send
Save
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (3)
Test Results
Status: 200 OK
Time: 24 ms
Size: 121 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1
17740


localhost:48080/api/v1/event/count
17740