Running on a RaspberryPi

This document will walk you through installing and running the essential EdgeX Foundry microservices on a RaspberryPi for testing and evaluation. This should not be used in a production environment. This guide is for developers who want to learn EdgeX by running it as a mock gateway device on a commonly available hobbyist device.

> **Know your device**
>
> This guide has been tested to work on a RaspberryPi 3B+, it may work on other models of the RaspberryPi 3, but it will not work on any older or lighter variations of the RaspberryPi as it requires a 64bit capable CPU.

Step-by-step guide

1 Install the OS
    1.1 Create swap file
2 Installing EdgeX
    2.1 From the snap
    2.2 With docker-compose
    2.3 From source

Install the OS

For this setup we will be using the 64-bit version of Ubuntu Server 18.04.3 LTS specifically for the Raspberry Pi 3B+.

Download the image file for Ubuntu Server from http://cdimage.ubuntu.com/ubuntu/releases/bionic/release/ubuntu-18.04.5-preinstalled-server-arm64+raspi3.img.xz then insert your microSD card into your computer so that you can flash the disk image onto it. If you are running Ubuntu or any other Linux distro, you can flash the image using this command:

> *xzcat **ubuntu-18.04.5-preinstalled-server-arm64+raspi3.img.xz** | sudo dd bs=32M of=**/dev/mmcblk0** iflag=fullblock oflag=direct status=progress; sync*

Note that you will need to replace the part in bold with the path to your own microSD card, and that it should point to the device and not a partition on the device.

After this is done, insert the microSD card into your Raspberry Pi 3B+ and power it on. You should see a screen with the Raspberry Pi logos at the top. Even though we are flashing a pre-installed image, there are still a number of things that need to be setup on first boot. However, after a few minutes, it should take you to the login screen. The default username is 'ubuntu' and default password is 'ubuntu'. You will be required to change the password on first login.

Create swap file

The RaspberryPi 3B+ has a full gig of ram, but while running MongoDB (and also when building from source if you choose to do so) this may not be enough. So before going any further you will first want to create some additional memory capacity by adding a swap file to your filesystem. Run the following command to create a new 2GB swap file, start swapping on it, and automatically remount the swapfile on reboot:

```
sudo touch /var/swap
sudo chmod 600 /var/swap
sudo dd if=/dev/zero of=/var/swap bs=1M count=2048
sudo mkswap /var/swap
sudo swapon /var/swap
sudo sh -c "echo /var/swap swap swap defaults 0 0 >> /etc/fstab"
```

You may also want to be able to connect and manage your device remotely, in which case you should also install:

> *sudo apt install openssh-server vim-tiny*

Installing EdgeX

There are three ways provided by the EdgeX Foundry to install and run the EdgeX services. We provide pre-packaged versions as a snap package (works on most Linux distros), as docker images, or you can build the EdgeX code from source and run the executables directly.

From the snap

The easiest way to get EdgeX Foundry is to install the officially supported edgexfoundry snap from the snap store.

To install the snap run:

```
sudo snap install edgexfoundry
```

If you want to install a particular release such as Delhi, you can use:

```
sudo snap install edgexfoundry --channel=delhi
```

The snap contains all of the default core-* services, support-*, export-* and security-* services though by default only the security-* and the core-* services are enabled in the snap. To enable other services (such as export-distro in this example), use snap set:

```
sudo snap set edgexfoundry export-distro=on
```

With docker-compose

EdgeX Foundry additionally supports running via a docker-compose file from the IoTech Community Developer Kit. To run that docker-compose file, install docker:

You can easily install docker via a snap:

```
sudo snap install docker
```

Or by following other installation methods here.

After that, download the docker-compose file using:

```
wget -O docker-compose.yml https://raw.githubusercontent.com/edgexfoundry/demo-grove-pi/master/docker-compose-demo-grove.yml
```

Then start the services with docker-compose

```
docker-compose up -d
```

From source

Note that running EdgeX from source is not a supported way and not recommended for production.

Install system dependencies

Once you have logged in, it's time to install the system dependencies needed to run EdgeX Foundry. Connect your device to your network and run the following commands:

```
sudo apt update

sudo apt upgrade -y

sudo apt install build-essential git wget libzmq3-dev pkg-config
```

Install Go & Glide

To get the exact same version of Go as used by the EdgeX, install it using the go snap (note that the project currently uses Go 1.11 for the upcoming Edinburgh release, but the Delhi release was originally built with go 1.10, however go being backwards compatible you should still be able to build the Delhi release of EdgeX with go 1.11) :

```
sudo snap install go --channel=1.11 --classic

cat >> ~/.bashrc << 'EOF'

export GOPATH=$HOME/go
```

```
export PATH=/usr/local/go/bin:$PATH:$GOPATH/bin
EOF
source ~/.bashrc
mkdir -p $GOPATH/bin
```

If you are going to build the Delhi release, you also need to install Glide, which is used to manage go dependencies for EdgeX Foundry. Install it with:

```
wget https://github.com/Masterminds/glide/releases/download/v0.12.3/glide-v0.12.3-linux-arm64.tar.gz
tar -C $GOPATH/bin -xvf glide-v0.12.3-linux-arm64.tar.gz --strip 1 linux-arm64/glide
```

Get the EdgeX Foundry source code

Now that you have Go and Glide installed, you can tell them to fetch the EdgeX services and their dependencies:

```
go get github.com/edgexfoundry/edgex-go
```

Then cd into the directory with the EdgeX source code:

```
cd ~/go/src/github.com/edgexfoundry/edgex-go
```

Building EdgeX Go microservices

There are two steps for building the EdgeX Go microservices, the first to prepare the build, and the second to compile it:

```
make prepare
make build
```

If you are going to build the Delhi release of EdgeX, install glide as above and then checkout that branch:

```
git checkout delhi
```

Install and setup MongoDB

EdgeX used MongoDB for local data storage. You can install it with:

```
sudo apt install mongodb-server
```

and verify that it's running with:

```
systemctl status mongodb
```

If it's not running you can start it with:

```
sudo systemctl start mongodb
```

Once it's up and running, it needs to be initialized with data for the EdgeX services, you can do that with the **init_mongo.js** file:

```
wget https://github.com/edgexfoundry/docker-edgex-mongo/raw/master/init_mongo.js
sudo -u mongodb mongo < init_mongo.js
```

Test run EdgeX services

Now that you have the EdgeX go services built and all the dependencies installed and running, you can run the EdgeX services themselves. The sourcecode contains a convenient script for doing this, in the same directory as your can make build above, run:

```
make run
```

This will start all of the EdgeX go services and leave them running until you terminate the process (with Ctrl-C). While it's running you can make EdgeX API calls to it using the IP address of your RaspberryPi.

Make EdgeX a system service

In order to keep the EdgeX services running when you're not logged in or connected to the RaspberryPi, and to have it start automatically when it boots, you can create a SystemD service to manage it. Create a new file at **/etc/systemd/system/edgex.service** with the the following content:

```
[Unit]
Description=EdgeX Foundry Microservices
After=network.target auditd.service
ConditionPathExists=/home/pi/go/src/github.com/edgexfoundry/edgex-go/bin

[Service]
WorkingDirectory=/home/pi/go/src/github.com/edgexfoundry/edgex-go/bin
ExecStart=/home/pi/go/src/github.com/edgexfoundry/edgex-go/bin/edgex-launch.sh
Restart=on-failure
RestartPreventExitStatus=255
Type=simple

[Install]
WantedBy=multi-user.target

Alias=edgex.service
```

Update the systemd configuration with:

```
sudo systemctl daemon-reload
```

and then you can start the EdgeX service with:

```
sudo systemctl start edgex
```

This will now be run every time you boot your RaspberryPi. You can verify that it's running with:

```
sudo systemctl status edgex
```

Related articles

- Badge Program
- Running on a RaspberryPi
- How to Run EdgeX on a Raspberry using Balena
- How to Access EdgeX Micro Services running on Kubernetes

Make run