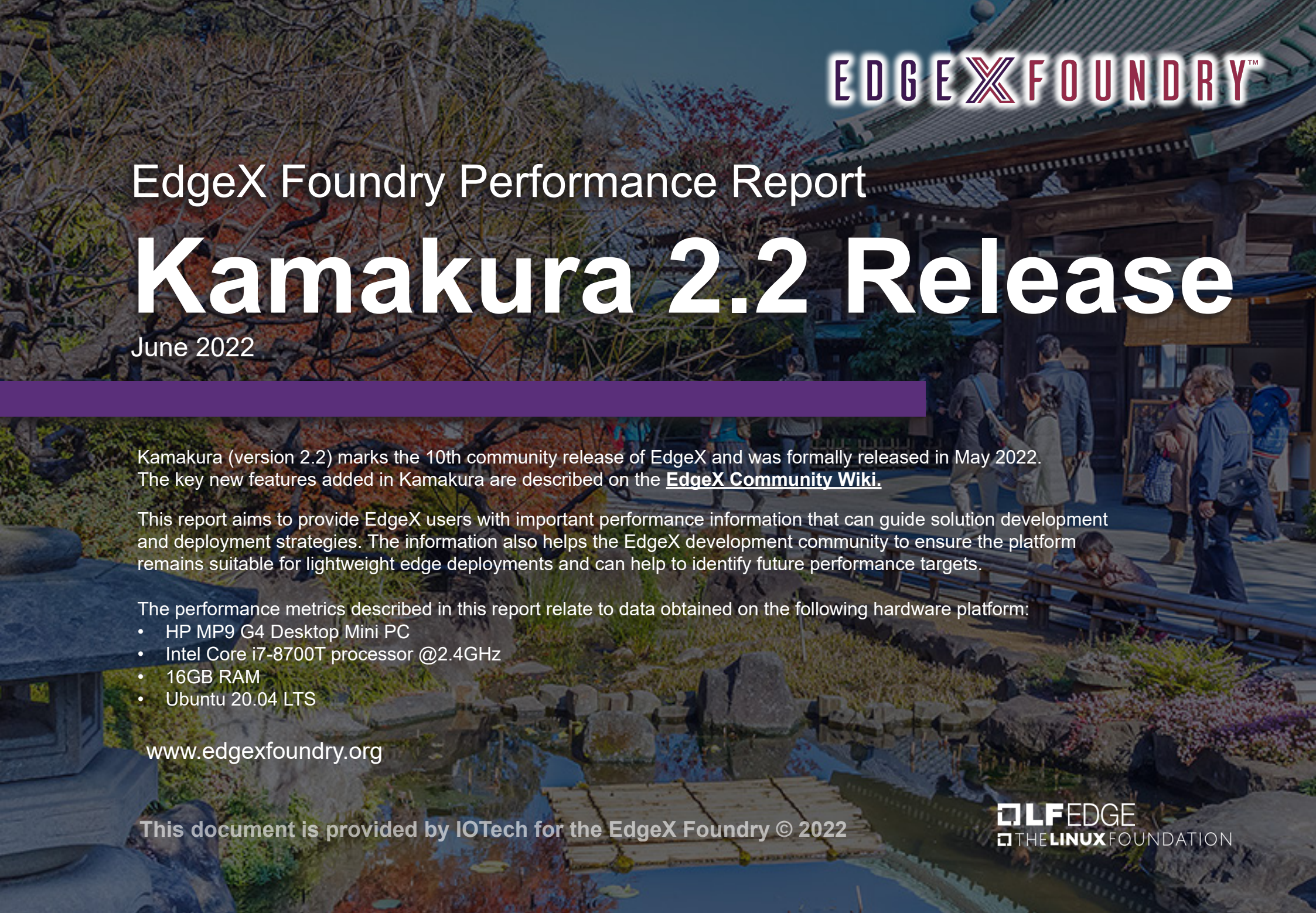


EDGEXFOUNDRY™

EdgeX Foundry Performance Report

Kamakura 2.2 Release

June 2022

Kamakura (version 2.2) marks the 10th community release of EdgeX and was formally released in May 2022. The key new features added in Kamakura are described on the **EdgeX Community Wiki.**

This report aims to provide EdgeX users with important performance information that can guide solution development and deployment strategies. The information also helps the EdgeX development community to ensure the platform remains suitable for lightweight edge deployments and can help to identify future performance targets.

The performance metrics described in this report relate to data obtained on the following hardware platform:

- HP MP9 G4 Desktop Mini PC
- Intel Core i7-8700T processor @2.4GHz
- 16GB RAM
- Ubuntu 20.04 LTS

www.edgexfoundry.org

LF EDGE
THE LINUX FOUNDATION

EdgeX Foundry Deployment Options

The EdgeX Foundry platform is a collection of modular microservices that each perform a specific role at the IoT edge. While some EdgeX microservices are used in almost all deployments of the platform (e.g. the Core Services), the specific set of microservices needed will depend on the exact requirements of each use case.

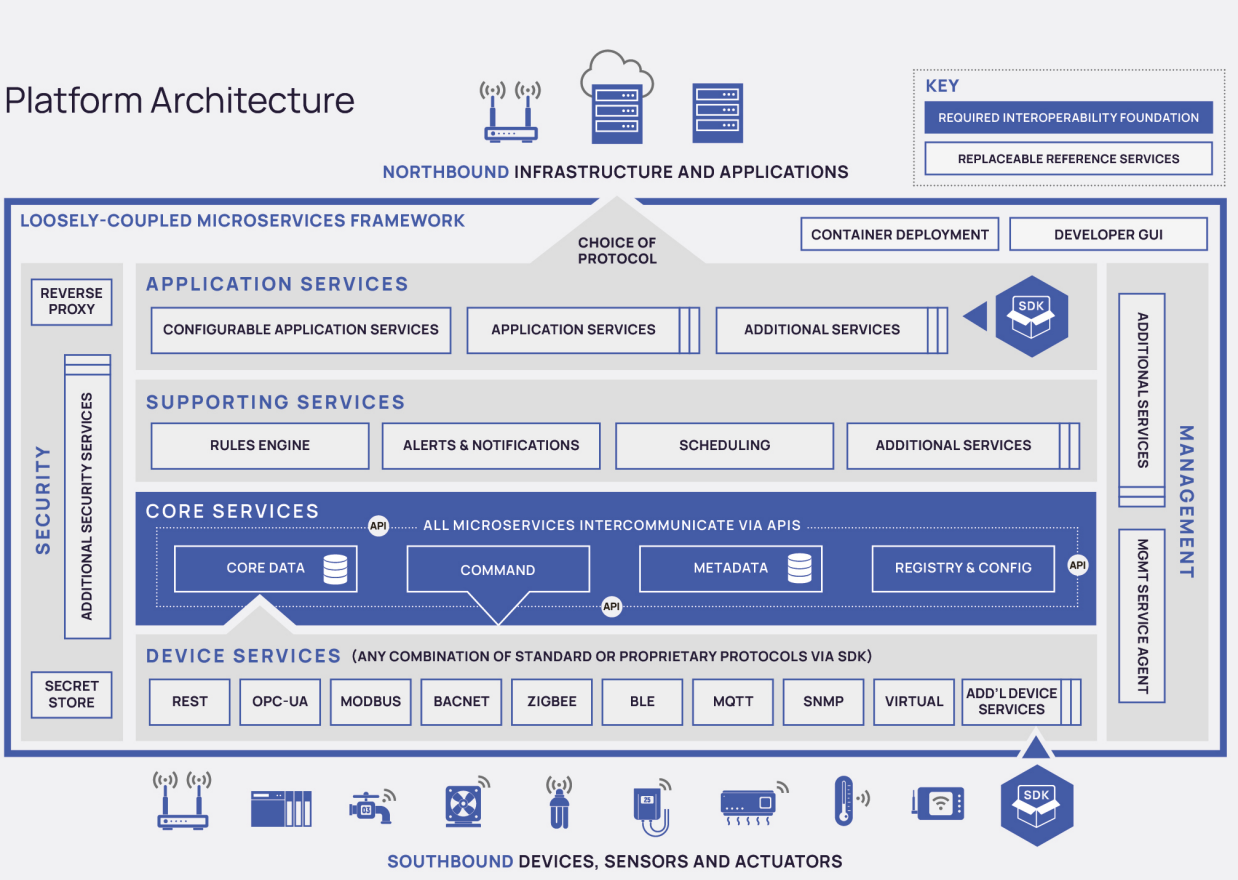
Since EdgeX consists of a set of required and optional microservices, this report provides performance data and resource usage relating to two EdgeX deployment classifications:

Typical EdgeX deployment

A typical EdgeX deployment provides all of the microservices as envisioned by the original architects. This includes microservices for ingesting data from different edge protocols, edge decision making, notifications and alerts, device actuation and streaming to a cloud or IT endpoint. A typical EdgeX deployment also includes the full EdgeX security services.

Minimal EdgeX deployment

A minimal EdgeX deployment provides only the microservices needed to ingest data from a single edge protocol and stream that data northbound to a cloud or IT endpoint. A minimal deployment may implement alternative security approaches to the EdgeX security services that are provided as standard.



The EdgeX Microservices

EDGE X FOUNDRY™

Microservice	EdgeX Deployment Name	Description
CORE SERVICES		
Core Data	edgex-core-data	Optional data store for readings collected by devices and sensors
Core Metadata	edgex-core-metadata	Used by other services for knowledge about the devices and how to communicate with them
Core Command	edgex-core-command	Optional invocation of devices on behalf of other services, applications or external systems
Database	edgex-redis	The default database, implemented via Redis
Configuration & Registry	edgex-core-consul	Optional centralized service configuration, implemented via Consul
DEVICE SERVICES		
REST Device Service	edgex-device-rest	For interacting with edge devices that provide a REST-based API
Virtual Device Service	edgex-device-virtual	For simulating device data
Note that many other Device Services are available both in EdgeX and provided by commercial vendors		
SUPPORTING SERVICES		
Rules Engine	edgex-kuiper	Reference rules engine implemented by EMQ X Kuiper
Support Notifications	edgex-support-notifications	Delivers notifications to inform of important system events
Support Scheduler	edgex-support-scheduler	Executes operations on a configured interval or schedule
APPLICATION SERVICES		
App Service Configurable	edgex-app-service-configurable	Application Service that can be configured to execute built-in transform and export functions
SECURITY SERVICES		
API Gateway	edgex-kong	Provides a single point of authorized entry for all EdgeX REST traffic
API Gateway Database	edgex-kong-db	Database required to work with the API Gateway
API Gateway Setup	edgex-security-proxy-setup	Service required to configure the API Gateway
Secret Store	edgex-vault	Central repository to securely store EdgeX tokens, passwords and certificates etc
Secret Store Setup	edgex-security-secretstore-setup	Service required to configure the Secret Store
Security Bootstrapper	edgex-security-bootstrapper	Provides secure activation of the EdgeX security services

EdgeX Memory Footprint

Each EdgeX microservice is typically implemented in either Go or C and then compiled into an executable which has a size or footprint as it sits on disk. For convenient deployment and orchestration, each microservice executable can also be built into a container image so it can be run as a standalone service with less dependencies on the host. This greatly improves portability and platform independence. EdgeX provides container images for both Docker and Ubuntu Snaps.

Both containerized Docker image footprint data and non-containerized executable footprint data is measured and shown below.

Microservice	Image Footprint (MB)	Executable Footprint (MB)
edgex-core-data	22.50	14.15
edgex-core-metadata *	17.01	11.33
edgex-core-command	16.07	10.39
edgex-redis *	32.37	N/A
edgex-core-consul	115.97	N/A
edgex-device-virtual	36.07	27.69
edgex-device-rest *	36.10	27.73
edgex-support-notifications	26.87	20.72
edgex-support-scheduler	26.29	20.61
edgex-kuiper	45.58	N/A
edgex-app-service-configurable *	36.43	27.56
edgex-kong	142.58	N/A
edgex-kong-db	205.16	N/A
edgex-security-proxy-setup	26.90	N/A
edgex-vault	186.43	N/A
edgex-security-secretstore-setup	28.51	N/A
edgex-security-bootsrapper	19.06	N/A
TOTAL	**1019.19 MB**	

Typical Deployment

The footprint for a typical EdgeX deployment using Docker containers is approximately **1020 MB**. This includes the configuration service, the command service, two device services, all supporting services and all security services.

Minimal Deployment

The footprint for a minimal EdgeX deployment using Docker containers is approximately **122 MB**. These services are marked in the table opposite with an asterix (*).

***Note**: The executable footprint is not recorded for third-party open source services (Redis, Consul, Kuiper, etc) so N/A is listed for those services.*

Please see the following page for mechanisms available to help further reduce the footprint of some of the EdgeX microservices.

Mechanisms to Minimize Memory Footprint

Note that EdgeX Foundry currently provides pre-built container images that provide full functionality for the user. The above footprint values, therefore, represent the full and flexible capability of EdgeX such as providing different options for the internal EdgeX message bus (e.g. ZeroMQ, MQTT and Redis Pub/Sub) and optional security features such as the new Delayed Start capability. However the nature of the Go language (in which EdgeX is largely implemented) is that Go libraries must be referenced in a Go Module file and therefore contribute to footprint size, whether they are actually utilized by the user or not during runtime.

Configurable Build Steps

Providing pre-built images containing all EdgeX functionality is simple and convenient for many EdgeX adopters. However, some users, particularly those targeting deployments on smaller gateways or other resource constrained devices, would prefer the option to conditionally choose which optional features to build into the EdgeX images.

See opposite for how EdgeX allows the Delayed Start functionality to be included or excluded in the EdgeX images.

Ongoing Footprint Considerations

An aim of the EdgeX Technical Steering Committee is to continue to improve and enhance EdgeX, but always be mindful of footprint considerations. As such, this approach of providing conditional build steps will continue to be utilized where possible when optional EdgeX functionality requires the additional of significantly large library dependencies. EdgeX adopters will still be able to rebuild the microservices to their own specific requirements, while commercial suppliers of EdgeX can provide their customers with pre-built microservice images that match their needs.

Example: Delayed Start

Delayed Start allows for services to be added and started at anytime and still receive security tokens without the need for a restart of the whole platform.

Analysis found that go-spiffe, the Go library that provides the Delayed Start capability, contributes to a footprint rise of approximately **15MB** for each EdgeX microservice. This can be significant especially when not every EdgeX use case requires the functionality.

An optimization of this EdgeX 2.2 Kamakura release is to provide the Delayed Start capability, by default, in only the EdgeX microservices where it is most likely to be utilized. The EdgeX Core Working Group concluded that Delayed Start should be included in the default pre-built images of the Device Services, Supporting Services and Application Services, but excluded from the default pre-built images of the Core Services. This is because the Core Services are unlikely to be started or restarted after the EdgeX platform is running.

However note that these are just defaults. The code has been rationalized such that EdgeX adopters are able to easily rebuild the EdgeX microservices to include or exclude the Delayed Start capability without major effort. Users wishing to rebuild the Core Services to include the Delayed Start capability functionality would see rises of around **15MB** per service. Likewise, users wishing to rebuild the Device Services, Supporting Services and Application Services without this capability would see a decrease of around **15 MB** per service.

See the EdgeX Secrets Module for further information on this feature.

EdgeX CPU Consumption

Each EdgeX microservice has its CPU consumption measured as it is started as a Docker container. The CPU usage is reported by the Docker engine and is measured as a percentage of the available CPU on the machine. In general, the measure of usage at startup is a good indication on the upper bound for many of the services. Note that the characteristics of different chip architectures may affect the CPU utilization.

Microservice	Maximum (%)	Minimum (%)	Average (%)
edgex-core-data	0.48	0.02	0.11
edgex-core-metadata *	0.12	0.01	0.04
edgex-core-command	0.07	0.01	0.03
edgex-redis *	0.63	0.11	0.25
edgex-core-consul	4.91	0.40	1.02
edgex-device-virtual	0.13	0.00	0.02
edgex-device-rest *	0.02	0.00	0.01
edgex-support-notifications	0.05	0.02	0.03
edgex-support-scheduler	0.13	0.08	0.10
edgex-kuiper	0.03	0.00	0.01
edgex-app-service-configurable *	5.57	0.00	0.60
edgex-kong	11.34	0.01	2.09
edgex-kong-db	0.06	0.00	0.02
edgex-security-proxy-setup	0.00	0.00	0.00
edgex-kong-vault	4.79	0.72	1.59
edgex-security-secretstore-setup	0.00	0.00	0.00
edgex-security-bootstrapper	0.00	0.00	0.00
TOTAL	**28.33**	**1.38**	**5.92**

Typical Deployment

Maximal CPU usage recorded for a typical EdgeX deployment using Docker containers on the test hardware is approximately **28%**. This includes the configuration service, the command service, two device services, all supporting services and all security services.

Minimal Deployment

Maximal CPU usage recorded for a minimal EdgeX deployment using Docker containers on the test hardware is approximately **6%**. These services are marked in the table opposite with an asterix (*).

EdgeX Memory Consumption

Each EdgeX microservice has its memory consumption measured as it is started as a Docker container. The memory consumption is reported by the Docker engine and is measured in Megabytes (MB). In general, the measure of usage at startup is a good indication on the upper bound for many of the services.

Microservice	Maximum (MB)	Minimum (MB)	Average (MB)
edgex-core-data	11.14	10.38	10.90
edgex-core-metadata *	9.06	8.63	8.75
edgex-core-command	7.94	7.50	7.75
edgex-redis *	3.18	2.65	2.95
edgex-core-consul	40.78	34.42	37.20
edgex-device-virtual	14.71	13.79	14.28
edgex-device-rest *	13.05	12.72	12.86
edgex-support-notifications	8.55	8.09	8.39
edgex-support-scheduler	8.53	7.81	8.34
edgex-kuiper	9.29	8.95	9.15
edgex-app-service-configurable *	14.15	13.36	13.81
edgex-kong	110.96	108.12	108.42
edgex-kong-db	18.46	18.24	18.36
edgex-security-proxy-setup	0.00	0.00	0.00
edgex-vault	121.23	121.10	121.11
edgex-security-secretstore-setup	5.15	5.15	5.15
edgex-security-bootstrapper	6.49	6.06	6.22
TOTAL	**402.67**	**386.97**	**393.64**

Typical Deployment

Maximal memory consumption recorded for a typical EdgeX deployment using Docker containers on the test hardware is approximately **403 MB**. This includes the configuration service, the command service, two device services, all supporting services and all security services.

Minimal Deployment

Maximal memory consumption recorded for a minimal EdgeX deployment using Docker containers on the test hardware is approximately **39 MB**. These services are marked in the table opposite with an asterix (*).

EdgeX Startup Times

The startup times are measured for each of the microservices developed by the EdgeX Foundry community, both with and without the security services enabled.

Startup times include any overhead associated with creating the Docker containers in which the microservices run. Note that the metrics are obtained while starting all microservices at the same time so any dependencies between the services starting are included in the data. The total time recorded is the time taken for all of the services to be started, rather than a sum of all individual startup times.

Microservice	Average with Security (s)	Average without Security (s)
edgex-core-data	19.91	9.88
edgex-core-metadata *	19.93	9.78
edgex-core-command	20.05	10.63
edgex-device-virtual	25.46	16.11
edgex-device-rest *	25.53	16.09
edgex-support-notifications	19.81	9.80
edgex-support-scheduler	19.62	9.82
edgex-app-service-configurable *	20.65	10.15
TOTAL	**25.56**	**16.13**

Typical Deployment

The average startup time recorded for a typical EdgeX deployment using Docker containers on the test hardware is approximately **25 seconds** This includes the configuration service, the command service, two device services, all supporting services and all security services.

Minimal Deployment

The average startup time recorded for a minimal EdgeX deployment using Docker containers on the test hardware is approximately **16 seconds**. These services are marked in the table opposite with an asterix (*).

Note also that services can of course be started and stopped as EdgeX runs. In this case it is not necessary to create a new Docker container for each service. Restarting an already-created Docker container can reduce the startup overhead.

EdgeX Operational Latency

Ping Response Times

The ping response times are measured for each of the microservices developed by the EdgeX Foundry community and represent the reactivity of each service when it receives an external HTTP/REST request. The tests are ran both with the services behind the API Gateway and then without security enabled.

Microservice	Request Latency (API Gateway Security)	Request Latency (No Security)
	Average Response Time (MS)	
edgex-core-data	9.46	1.05
edgex-core-metadata	9.46	1.06
edgex-core-command	9.53	1.06
edgex-device-virtual	9.55	1.02
edgex-device-rest	9.56	1.07
edgex-support-notifications	9.49	1.05
edgex-support-scheduler	9.42	1.07
edgex-app-service-configurable	9.42	1.03

Ping response times recorded on the test hardware are consistently around **9-10ms** when the API Gateway is included. Without the API Gateway, response times are consistently around **1ms**.

Data Export Latency

Data export latency represents the time it takes to collect EdgeX data from the “southside” devices & sensors and deliver it through the platform to the “northside” applications.

The latency reported here relates to the Virtual Device Service, which simulates 3 different data attributes. Each Device Service type (Modbus, BACnet, etc), will obviously have its own data rates associated with collecting the data from the physical edge devices or sensors, but the Virtual Device provides useful indicative performance data. It is planned to provide performance relating to more Device Service types in future versions of EdgeX.

Virtual Device Attribute	Average with Security (ms)	Average without Security (ms)
Random-Integer-Device	4.1	4.6
Random-UnsignedInteger-Device	4.5	4.2
Random-Boolean-Device	2.4	2.5

The Virtual Device data is delivered through the EdgeX platform on the test hardware at an average latency of approximately **2-4 ms**. There is no discernable latency overhead when EdgeX security is enabled.

EDGE X FOUNDRY™

Test Methodology

All EdgeX microservices were run with their default configurations. In order to provide accurate and reliable statistics the tests were ran as follows:

- The CPU and memory consumption metrics were gathered by obtaining the data 10 times at an interval of 7 seconds between each measurement
- The startup time data was obtained by starting the EdgeX microservices 5 different times
- The ping response test was performed 100 times for each EdgeX microservice
- The data export latency metric was performed 10 times for each of the Virtual Device attributes

The full raw performance metrics, as well as data for other hardware platforms, is provided on the **EdgeX Community Wiki**

Contact Us

For general information about EdgeX Foundry, or membership inquiries, please email **info@lfedge.org**

Visit our website at **www.edgexfoundry.org**

IOTech

This document was produced by **IOTech Systems**.

IOTech is The Edge Software Company with products that include **Edge Xpert** a value-add and commercially-supported implementation of EdgeX Foundry. Key features include a large selection of industral grade device service connectors including Modbus, BACnet, OPC UA, BLE, etc.

IOTech also provides lightweight and real-time extensions to EdgeX as well as edge management orchestration software.

If you have questions about this report or if you need more information on IOTech or Edge Xpert, email **info@iotechsys.com** or visit the website **www.iotechsys.com**

LF EDGE
THE LINUX FOUNDATION