California Release

NOTE: California has been released. See the release page for details of this release.

- NOTE: California has been released. See the release page for details of this release.

Delivery: ~June 2018

The main theme for the California release of EdgeX is to provide a solid open source foundation for commercialization and deployment in a wide variety of IoT edge use cases.

Key to this is the first implementation of security infrastructure. Another key theme for the California release is improving overall performance and lowering the baseline footprint of the code base. California offers drop-in alternatives for all EdgeX microservices based in Go Lang.

Release Themes and Objectives

- Deliver initial security infrastructure (e.g. reverse proxy, key management)
- Deliver on promise for a performant, reliable IoT edge platform
- Reduce overall footprint by an order of magnitude through alternative microservice implementations in Go Lang with future services (especially device services) in languages like C/C++
- Enable near real-time performance (see targets below)
- Improve and overhaul the documentation set
- Blackbox tests for the entire EdgeX API set
- Arm 64 native testing - with continuous integration processes extended to produce artifacts and support the native testing

General Release Tasks and Notes

- Expand hardware support - ARM 64
  EdgeX has been built to be cross platform. Some micro services (such as those written in Go) require compiling to the native platforms (cross compiling). Once the program artifacts for the platform have been created, they also need to be containerized (example: in Docker container). Generally, the containerization is platform dependent, but there are exceptions. Finally, it is also desired that the micro services be tested on their native platform. While virtual machines and simulation environments can allow functional tests to occur, some issues and performance metrics for that platform can only be identified when running on the target platform. This task therefore includes:
  - CI process that includes cross compiling for Intel/Arm platform artifacts
  - Micro service Docker container images produced for Intel and Arm platforms
  - Native platform testing of the micro services on Intel and Arm platforms
- Improve and overhaul the documentation set
  Moving developer documentation from Wiki to GitHub so that it is updated/maintained/reviewed like code is (through formal pull requests, etc.). This allows the documentation to also be versioned with the code-base. Use of standard documentation tools and processes, allow the documentation to be “built” by the CI processes and released in more user friendly (and portable) fashion.
- Blackbox tests for the entire EdgeX API set
  Prior to the California release, there have been blackbox REST API tests against the core services (core-data, core-metadata, core-command). Additional blackbox REST API coverage is being added for the supporting services (logging, notifications, scheduler, etc.), export services (export-client, export-distro), and device services (some of which will apply to the SDK and can be used against generic device services built by the SDK).
- Meet initial performance targets
  While more requirements from the field are needed, the overall performance targets for EdgeX are listed at the bottom of this page and include running all the core, support and application micro services in < 1GB or RAM on a 64 bit CPU, requiring less than 32GB of disk storage space, start (collectively) in under 1 minute and actuate from sensor collection to device trigger in < 1 second.

Core & Supporting Service Tasks and Notes

- Convert all core and supporting services to Go Lang. Many of the services have been refactored since the California Preview so as to align better with Go Lang patterns and practices.
- Upgrade of Consul (from 0.7 to 1.0)

Export Service Tasks and Notes

- Convert export services to Go Lang.

- In addition to the previously existing HTTP/HTTPS, MQTT/MQTTS, Azure IoT Hub, and Google IoT Core “north side”export capability, additional north side connectors are provided to include XMPP, ThingsBoard, and Brightics IoT

Device Service SDK and Device Service Tasks and Notes

- While new device service SDKs (and associated device services) did not make this release, requirements and design for Go and C based SDKs are to be completed. New SDKs and device services will be released between California and the Delhi release as they become available.

System Management Task and Notes

- Provide a preliminary set of requirements and general roadmap for system management (or micro service management) to guide implementation in future releases.

Security Tasks and Notes

- Institute an OS reverse proxy (Kong) to protect the micro service APIs from unauthorized outside requests.
    - This will be for HTTP/S only (meaning security for MQTT and other protocols will not yet be provided)
    - With no (or very few) changes to existing micro services at this time
- Data Protection Services via HashiCorps’s Vault is integrated with EdgeX
  This will provide Key Management/secret storage for EdgeX going forward (to include certs, passwords, etc.)

DevOps Task & Notes

- Blackbox testing is automated with error results provided to interested parties.
  This feature helps to maintain the EdgeX APIs over time and insure that changes to the code base do on effect the micro service interfaces.
- Automated ARM 64 builds, testing and artifact production have been instituted on 3rd party platforms

Target Performance

- The target is to run all of EdgeX (all the services to include the virtual device service) on a Raspberry Pi 3 (1 GB RAM, 64bit CPU, at least 32GB storage space, running Raspbian Linux distro)

    - This is not an endorsement of the platform as our favorite or otherwise endorsed platform
    - It only suggests the general characteristics of a “developer community” target platform
    - This may not be entirely feasible without all Go replacements, but is the target and the development community will report back when /where this is not possible
    - For example, it is unlikely the target security implementation will fit on this size platform
- Additional “developer community” targets
    - Startup in 1 minute or less (post OS boot – from micro service start to all services up and available)
    - Throughput for one piece of data (with one IP device connected by hard wire – versus WiFi) from data ingestion at the Device Service layer, through Core Data, to Rules Engine and back down through Command Service finally to Device Service for actuation will be < 1 second