Hanoi Release

Potential Features and Tasks

This is a list of potential features and tasks for Hanoi. Nothing in this list should be considered in scope for Hanoi at this time. Some items are actually contradictory to one another.

3/22/20
Last Updated 3/26/20

- Geneva item that did not get completed
- Backlog or Hanoi roadmap item
- WG discussed Hanoi release item
- Kanban board

General / Cross Cutting Concern

2.0 release???

New feature

- Go 1.15?
- Message bus between select services
  - Requiring broker or going brokerless
  - Supporting synchronous, asynchronous or both
- Support running a service on a different host than the other services. Specifically making it easier to run a device service on a different host that the rest of Edge. As a stretch to this goal, allow for select service high availability.
- Improving EdgeX resiliency in face of issues or non-availability of some resources/services/etc. (typically for core and above services and not device services)
- Insure all micro services follow 12 factor app methodology (see https://12factor.net/)
  - Allow services to be load balanced
  - Allow services to fail over
  - Allow for dynamic workload allocation
  - Allow services to live anywhere and be moved without lots of configuration to be changed in other services
  - Allow services to be distributed across hosts - and across API gateways (requiring service to service communication protection)
- Support truly distributed microservices
  - Allow services to run on multiple host machines
  - Secure distributed EdgeX with reverse proxy
  - Cross EdgeX instance command actuation (ex: device X on EdgeX box A triggers action on device Y on EdgeX box B)
  - Front a collection of duplicate microservices with a load balancer (allow for the microservice copies to scale up or down based on load); allow multiple instances of any microservice (for future load balancing and scaling efforts - today only single instances are allowed)
- Develop a test environment/playground to test high-availability and distribute service functionality.

- Allow for new category of micro services: “Sharing Services” for East/West data exchange with non-Edgex entities
- Meetup support
- Hackathon kits/support

Tech Debt

- V2 API
- Deprecate Mongo
- Removal of Value Descriptors in favor of data typing in the reading
- Implementation of generic error handling across services
- Develop process for security vetting of 3rd party components
- Restructuring our compose files to take advantage of compose file overrides, which removed the duplication in all our compose files. See https://devilbox.readthedocs.io/en/latest/configuration-files/docker-compose-override-yml.html

Core / Supporting

New feature

- Allow for device hierarchy in metadata model: a device could be a manager for another device while also collecting data itself. Sending a command to a managing device could mean sending a command to all associated devices.
- Command: in order to protect the device from harmful commands, there should be the possibility to set a Min and Max limit for the value that is accepted on every single command. in fact the command service today is rather a hollow simple proxy, but in the future we very much envisioned adding additional security, caching to avoid having to hit the DS when unnecessary, and even grouping command requests for better resource conservation (especially for devices like BLE that get woken up when you hit them).
- Drop logging service
- Change core-data to support data
- Support for alternate logging formats and/or more structured logging
    - XML, CSV vs JSON
- Support automatic migration of Devices between Device Services
- Core Command support send commands to devices based on "label"

Tech Debt

- Tests should run cleanly when passed the "-race" argument
- Tests should run cleanly when passed the -race option
- Add hooks for hardware-assisted protection of Vault Master Key to security-secretstore-setup
- Streamline proxy certificate upload flow
- core-command coupled to core-metadata database
- Restructure the Redis DBClient implementation for events

Application Services / App Functions SDK / Analytics Integration

New feature

- Kuiper – next release

    - Decouple from EdgeX
- Deprecate Drools rules engine
- Support Cloud Event export
- Support additional northbound endpoints and protocol types. Examples include:
    - Tencent
    - Alibaba
    - IBM Watson
    - IoTivity
    - SAP HANA
    - DDS
    - AMQP
- Support enrichment functions (an EAI concept) in export services (or application services). Allow additional data or information to be added to the flow of sensor data to the northbound side. This might be information about the sensor/device that captured it or information about the commands to actuate back down on a sensor/device.
- Integrate to edge software/agents
    - AWS Greengrass
    - Microsoft IoT Edge
- Support additional northbound formats
    - Haystack
    - OPC UA
- App Service Configurable – Allow same function twice
- Fork the pipeline?
    - Allow various paths for different sensors for example
- Support multiple topics in SDK - go mod messaging
- Multiplexor to split out data from device services
- Pipeline per topic
- 0MQ export function

Tech Debt

- Improve binary data support
    - Local edge analytics may be fed binary data and create intelligence from it to allow for actuation at the edge based on the binary data (example: examine an image an detect the presence of a person of interest).
    - Support alternate message formats for service-to-service exchange (Protobuf, XML, etc.)
- Update SDK to use new persistence service

Device Services / Device Service SDKs

New feature

- Support Cloud Event import
- Data filter design between DS and Core Data
    - Provide a design about how to implement this before implementing.
    - If possible, can the filter functions be shared across App Services and D.S. (w/ App WG)
- Support additional southside connectors

    - Profinet/Profibus
    - CANBus
    - LORA
    - IoTivity
    - Zigbee
    - ZWave
- Better support mesh network protocols (Zigbee, BACNet, etc.) where devices know and sometimes manage other devices
- Downsampling: It is mentioned that the device service may receive from the device new unattended readings (e.g. in a pub/sub type of scenario). In this case, there should be a setting to specify whether we accept all readings or we decide to downsample because the source is pumping data too fast. This is actually a very common scenario when you deal with high frequency sensor packages.
- Bound checking
    - Number of operations that can be done
    - Max request size (that lends to DoS, etc.)
- Cache fairly static information (device profiles, device information, etc.)
- Allow for easier device removal

Tech Debt

- SDK alignment – can / should the DS and Application Functions SDKs be more aligned (design, usage, etc.)?

Security

New feature

- Create a hardware secret storage design
    - HW secure storage abstraction layer
    - How to protect the Vault Master Key
- Ensuring the services running are those expected and authorized (w/ DevOps assistance)
- Enable Vault PKI secrets engine
- Enable user-specified Kong proxy certificate
- Secret store unsealing daemon
- Configure TLS encryption for kong+postgres channel
- Secure service to service communications

Tech Debt

- Create and use a per service Vault token in the security services
- Blackbox tests of APIs through the API gateway
- Design work
    - How to implement HTTPS in EdgeX (that is, how to protect all service endpoints with HTTPS)
    - How to implement role-based security across our all EdgeX services.
- Eliminate remaining dependency on curl and jq in scripts/security-proxy-setup-checker.sh
- Make docker container images read-only
- Set no-new-privileges option on containers

- Develop process for security vetting of 3rd party components
- Make docker container images run as non-root users
- Streamline proxy certificate upload flow
- Secure Kong admin port with TLS
- Enable CORS for API Gateway
- Rewrite security-secret-setup to remove legacy cruft
- Pluggable random database password generation
- Add method to retrieve metadata inserted by file-token-provider
- Check the consul/registry for the configurable settings in security-secrets-setup

System Management

New feature

- CLI improvements (do we have idea of what these might be)
- Metrics collection per ADR 0006
- System management - storing system metrics locally
- System management - actuation based on metric change (a "rules engine" for control plane data)
- System management - add alerts and notifications (service down, metric above threshold, etc.)
- SMA support for other control plane protocols
- Helm charts or other Kubernetes “facilitation”

Tech Debt

- Improving system management capabilities to include providing an asynchronous set of APIs, offering more EdgeX specific APIs, storing management metrics (for store and forward) and exporting management data to 3rd party systems
- Look at Driving "Default Services List" via Configuration
- Refactor to enable operations against agent via executor
- Capture CPU metrics data for macOS

DevOps

New feature

- Sharpen our use of SonarCloud and provide developer education around it
- Automatic code formatting in CI/CD pipe
- Include a data filter between DS and Core Data (align with and share App Service filter function if possible)
- Produce deployment artifacts that contain multiple services (e.g. a single core service docker container versus core, metadata and command services)
- Produce service executables that combine services (e.g. create a single core executable that is core, metadata and command all in one) via the build/make process.
- Set up codecov for edgex-global-pipelines

Tech Debt

- Improve performance of pipelines
- Snap build improvements

    - Snap builds should use unbuffer for better output logging
    - Update snapcraft inside docker to use new setup from snapcraft
    - Make stage-snap jobs more smart about pushing metadata
- Root cause analysis of the ARM failures for blackbox-testing - Infrastructure failures

Certification

??

Vertical Solutions

??

Test / QA / Documentation

New feature

- User guidance on platform needs
    - More performance statistics
    - # of devices/per recommendations
    - Providing EdgeX users guidance on platform needs, sensor data throughput and deployment based on performance metrics. Specifically, with the Geneva performance testing apparatus, the EdgeX community will be able to answer these questions for the user:
        - Will EdgeX fit on my system? - size of EdgeX services, infrastructure, etc. and hardware/platform requirements
        - What is the speed of data through the system? - from device service sensor data ingestion to the rules engine and back down through command to another device service to trigger a put command, how long does this take?
        - How many "things" can be processed at a time? – with caveats on the type of thing, type of data, etc.
        - These questions need to be answered on real hardware (both Intel and ARM)

Tech Debt

- System integration / interoperability tests - Device Service read data -> Core Data -> Rules Engine or Application/Export Service -> Command
- Add unit tests/testing for global libraries. (w/ DevOps help)
- Blackbox tests to edgex-go repo?
- Blackbox tests against snaps
- Configuration testing – testing non-default and dynamic config changes

Open Horizons

- Should be a separate project in spring 2020.
- Roadmapping/direction is self determined
- Any additional next steps for EdgeX integration?