General

- Anointing a new Release Czar to better manage the Fuji release
- Move API docs from RAM to Swagger
- Move to Go 1.12 (minimal, perhaps Go 1.13)
- Use nanoseconds for Event/Reading creation time
- Update the EdgeX "Offerings" page
- Reconstitute the EdgeX marketing working group
- Research options for better building/packaging/using alternate infrastructure elements
- Capture unit, integration, and other testing coverage metrics so that the test coverage can be prepared at each face to face meeting.

Application Services / Functions SDK

- Archive Export Services
- Use of Vault to store appl service secrets (like cloud keys, tokens, passwords, etc.)
- Provide configurable application service
- Add following connectivity endpoint options: MQTTS, HTTPS
- Add following connectivity endpoint examples: Azure IoT Hub, Amazon IoT Core
- Add Encryption and Compression functions
- Stretch goal to add Google Cloud IoT Core and one Chinese cloud provider

Supporting Services

- Research rules engine replacement options
- Explore OpenJDK support/update for rules engine JVM

Core Services

- Update Consul to 1.4
- Refactor to provide better decoupling and facilitate easier unit testing
    - Targeting 35% unit test coverage for core/support services for Fuji
- Add unit tests to get coverage percentage above TBD %
- Address current typing issues that are not helpful to adding more unit testing (more encapsulated types for tests)
- Setup better transactional boundaries (ex: update of device profile)
- Move Value Descriptor management out of Device Service SDK and into Core Services
    - Addressing issues around Value descriptor & Device Profiles out of sync when changes occur

Device Service / SDK

- Add blackbox tests (defined by a test plan) to the SDK (and thus device services).
- Create a generic IP camera device service (using ONVIF protocol where possible)

- Provide dynamic/automatic discovery scaffolding in the SDKs to allow device services to automatically discover and provision new devices at the DS creator's discretion (provision watcher work)
- Stretch goals
    - Allow for the deregistration of devices/device services
    - The SDKs will implement a means to provide a cache of readings. This allows the collection and response for a request of a reading to be decoupled (and more asynchronous).

Security

- Allow services to get their secrets (tokens, passwords, certificates, etc.) from Vault (Carried over from Edinburgh) (ongoing)
- Move API gateway and secret store code to edgex-go (requires code review and approve)
- PKI infrastructure will be added to generate the tokens and keys necessary to use Vault, Kong and other security services (today, EdgeX relies on the keys to be generated elsewhere and then used by security apparatus) (Intel, in their private git, no further insights except that they are working on it)
- EdgeX micro service secrets will be stored and distributed per service in Vault using namespaces (today, all services access the secret store with the same key and therefore have access to all secrets)
- Include technology to ensure services running in EdgeX are those expected (and authorized). (Malini created https://github.com/edgexfoundry/edgex-go/issues/1518 to elicit input from devops and community.)
- Define/design a hardware secure storage abstraction layer that will include a software implementation and allow platform providers to build hardware root of trust implementations that can be used by EdgeX to protect the Vault Master Key and other fundamental system secrets used at bootstrap time. (Ongoing .. to be covered at Aug 1/2019 F2F hosted by Intel, Bryon and Jim to share their design and status. Tingyu and I would prioritize PKI work above this)
- Add documentation defining, at a higher level, what security features EdgeX offers and what is the security feature roadmap of EdgeX. (TODO: list all our security work in wiki)
- The community will renew/refresh a threat assessment.(Tingyu will set up a private git repo and capture his thoughts and work on this. Time to revisit Bryon Nevis/Intel documents on threat model too)

System Management

- Refactor the SMA executor to accomplish Start/stop/restart tasks by the executor (to include stop/restart of SMA).
- Refactor metrics collection - moving metrics collection to the executor so that it can remain platform and even service implementation agnostic.
- Add the ability of the SMA to set configuration (when the configuration is writable).
- As a stretch goal, add an SMA Translation layer (like LWM2M or SNMP)

Certification

- Provide self-assessment of device services by EOY 2019 (with a stretch goal of self-assessment of all services).
    - Task dependent on device service black box tests by Fuji

Test/QA

- Increase performance metric capture from all services. The goal is to, by the Geneva release, be able to answer 3 primary performance questions:
    - 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.
- Improve blackbox test structure including reorganization of the tests and better test case documentation.
- Create a new test framework (e.g. Robot or Cucumber) to support additional types of functional/blackbox and system integration tests (e.g. Device Service or system level latency tests).
- Remove documents from the other code repositories to its own repository
- Add performance testing automation. Specifically:
    - Automate API Load testing (measure response time) and metrics (CPU, memory) collection for all EdgeX micro services
    - EdgeX micro service startup times
      Improve the documentation to address some critical and re-occurring needs: How to get started with Windows and a common troubleshooting guide
- Add a documentation versioning tool (like Sphinx)

DevOps

- Add static artifact analysis into the EdgeX Jenkins Pipeline (analysis of Docker /runtime artifacts, not the source code)
    - Clair Server – issues on where to run; may have to defer to Geneva
- Add code and artifact signing with semantic versioning
    - Gitsemver along with lftools
- Conduct build performance optimizations by:
    - Adding Pipelines for EdgeX Foundry base build images
        - Jenkins build performance optimizations for base build images completed.
    - Allow base build images to be managed locally within Nexus
    - Leverage PyPi Proxy for local pip dependencies
- Explore static code analysis like Coverity, Checkmarx, GuardRails, Synk, SonarQube

Vertical Solutions

- Consider partial/full integration of CNCF CloudEvent as Core Data Event v2, or import/export services
- Consider Telegraf Device Service
- Consider investment in Core Data / meta data / schemas
- Consider configurable data pipelines
- Retail: Real-time transaction log device services
- Retail: VMS/ipcam device services
- Retail: Video Inference device/app services