DevOps Working Group

Thursday July 18, 2019
# Agenda

<table>
<thead>
<tr>
<th>Time</th>
<th>Topic</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 min</td>
<td>Edinburgh: Dot Release Discussion / Planning 7/22</td>
<td>James Gregg / Eric Ball / Jim White</td>
</tr>
<tr>
<td>10 Min</td>
<td>Fuji Update</td>
<td>James Gregg</td>
</tr>
<tr>
<td>10 min</td>
<td>Performance Testing – LF Configuration EB IAC</td>
<td>Emilio</td>
</tr>
<tr>
<td>10 Min</td>
<td>Clair – Docker Image Scan Reports / Demo</td>
<td>Ernesto</td>
</tr>
<tr>
<td></td>
<td>Opens – Nexus Questions</td>
<td>All</td>
</tr>
</tbody>
</table>
Attendees
EdgeX DevOps WG Update (Edinburgh Dot Release)

Edinburgh Dot Release (1.0.1)

- List of Artifacts impacted
  - go-mod-messaging (this is the module that doesn’t get released).

All other services need to be released (1.0.1)

- 1.) We need to merge a fix into the master and Edinburgh branches of go-mod-messaging
- 2.) The Edinburgh branch of both edgex-go and app-functions-sdk will need to consume the Edinburgh branch of go-mod-messaging
- 3.) We will need to republish the artifacts for all services in edgex-go for the Edinburgh release as well as app-functions-sdk
- 4.) We will need to adjust the Edinburgh docker-compose files to utilize the version numbers applied to the new images by the above actions.

Decision: TSC Meeting voted and Approved on 7/17
Planned Release now scheduled for 7/22
DevOps WG Update

• Fuji Scope
  • Container Scanning (Clair Server landing request) has now been committed to by Linux Foundation with resources planned and funding approved for hosting on AWS.
    • Progress made on Clair reporting - Demo and Discussion in DevOps WG
  • Codecov.io integration completed on all repos with tests
  • Static Code Analysis Tools - decision to defer decision on SonarCloud
    • Plan to review analysis completed to date in the Security WG

• GitHub repo audit completed and actions taken to address empty repos
  • Next steps: audit Teams of Committers
• Review of the Linux Foundation Infrastructure as Code re: Performance Testing completed
• Release Czar Proposal to use GitHub Projects for all Issues as an aide for the release mgmt.
# Work Review

<table>
<thead>
<tr>
<th>Helpdesk Ticket #</th>
<th>Description</th>
<th>Details</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>75648</td>
<td>Dedicated Clair server for EdgeX</td>
<td>Pending decision on strategy for K8s + cost / availability of resources with LF</td>
<td>WIP</td>
</tr>
</tbody>
</table>
Backlog Review
Analysis / Review of LF TIG IAC

- QA-Test WG reporting missing data when querying Grafana
- Issue is inconsistent between test runs
- Analysis of logs reveal
  - File corruption
  - Out of memory
  - “Influxdb Starting” which suggests container crashed and was spinning back up
- Emilio / Jacob reviewed the Terraform (infrastructure as code)
- Requested input from LF as an opportunity to vet our assumptions

Assumed AS-IS architecture

- Amazon Elastic Beanstalk
- Load-balancing, Autoscaling Environment
  - Instances: Min: 3, Max: 5
- Multiple Availability Zones
- Leveraging EFS storage for DB
- T3.Medium Instances, 1.5GB/Container

Recommendations

- Switch to single instance
  - InfluxDB OSS does not support clustering. HA for open-source requires custom layer to facilitate replication of data to all InfluxDB nodes
  - Switch to Elastic Beanstalk Single-Instance Environment
- Scale up single instance
  - InfluxDB sizing recommendations for AWS:
    - [https://docs.influxdata.com/influxdb/v1.7/introduction/Installation/#hosting-influxdb-oss-on-aws](https://docs.influxdata.com/influxdb/v1.7/introduction/Installation/#hosting-influxdb-oss-on-aws)
  - Minimum of 8GB RAM
  - Use R4 or M4.large
  - Vendor recommends SSDs
- Leverage EBS volume for InfluxDB data storage instead of EFS
  - Locking on network mount problematic

Nice To Haves

- Read-only access to dashboards to monitor performance
  - Elastic Beanstalk environment health console
- Access to CloudWatch logs
Clair Docker Image Scan Reports

Use of Clair vulnerability scanning within the EdgeX pipelines.

- Two approaches:
  - Embedded within pipelines. When images are pushed to nexus, a scan occurs and an HTML report is archived with the build.
  - Continuously scans via audit reports, part of the new ci-audit-pipelines repository.
- Results will be archived in Jenkins.
- [ASK] Should results be actionable or reporting only.

Geneva scope

- [ASK] Potential Slack integration when vulnerabilities are discovered.

Demo & Discussion
Use of Nexus for Test Automation Framework

• Seeking input on use of Nexus as an artifact management solution for test reports

• As part of the QA/Test, we are working on to introduce TAF which is based on robot/python.

• The artifacts (reports, logs, snapshots etc) generated as part of the test execution has to be stored in some repository and to be published.

• Looking for a way to utilize the existing Nexus infrastructure
  • We may need to create a new repository
  • To begin with, we need a way to experiment in Sandbox environment.
Meeting Minutes

Analysis of LF IAC of EB implementation on AWS
• After review of the recommendation, Cloud suggested to try the recommendations on the current AWS implementation
  • Decision: Work with LF to review the assessment, collaborate, and try the recommendations if acceptable to the LF Infra team. Will try to coordinate via existing JSD Ticket #

Clair Reporting
Decision: nothing actionable for now – just the reports
We will add a user story to the backlog for Slack integration (Geneva) but if possible pull it

Nexus for Test Automation Framework Reports
Need to submit a ticket so that the Nexus repo can be created and settings updated on Jenkins Sandbox
Edinburgh Retrospective

What went right?

• Communication of when the release was scheduled, was very clear.
• LF and DevOps team cooperating together seemed to work well but with pressure added on top of everyone
• Most artifacts were ready to release in the beginning. It didn’t seem like every repo was affected with issues (no extra work)
• Edinburgh Staging view was very helpful
• Great communication and collaboration between Intel DevOps team members and great prioritization
• Prioritization and Organization of the work (assignments and splitting up the work early on in the code release) was helpful

What went wrong?

• EdgeX-UI repos were late code drop
• Lack of a UI WG
• Communication around details was lacking from WG leads
• Need clear definition and understanding from WG leads that have different / independent release cycles
• JSD was introduced in the middle of the release and introduced issues that impacted communications with LF RE
• Availability and Competing priorities of Eric Ball / RE impacted release work
• Branches cut early caused extra work for both developers and DevOps
Edinburgh Retrospective (continued)

What Ideas would help next time?

• Jenkins Pipelines branch defined in Jenkinsfile
• Do Not cut the branch early
• Release from master
• Don’t allow PRs to master during code freeze (unless bug fix)
• Release Czar manages the release and acts as coordinator
• Need better visibility as to WIP during release. What does DONE look like?
• Set clear expectations for FUJI ahead of time
• Have a solid and well defined scope for executing the release.
• Shorten the release timeframe
• Actually Freeze Code
• Don’t pull in late code drops

What Actions will we take?

• Better coordination with RE to lay out the scope of work
• New UI WG - DONE
• Better tooling Pipelines (Geneva)
• Better clarity and organization / coordination of the work
• Release Czar coordinates and runs the release
• Shrink the release date where no development is going on during “code freeze”
• Create automation on the list of artifacts to release
Fuji Planning

Scope Discussions
Fuji – DevOps

**In**
- Static code analysis tool identified and integrated into the EdgeX Jenkins Pipeline for Docker image scanning (Clair Server)
- Explore SAST for true static code analysis to include additional tooling such as Fortify / Coverity
- Code and artifact signing with semantic versioning
- Fix Documentation – edgex-go
  - Create a new repo for edgex-docs
- Build Performance Optimizations
  - Pipelines for EdgeX Foundry base build images
  - Basebuild images managed locally within Nexus
  - Leverage PyPi Proxy for local pip dependencies
  - ARM builds – optimization leveraging different high CPU build nodes / OS (ARM Team)

**Out**
- Alternate deployment/orchestration
  - Beyond Docker/Snaps
  - Kubernetes
  - Kata Containers
  - …
- SonarQube – SonarCloud is already in play in the LF
  Decision: wait to see what codecov.io offers
- Suggestion to rename all of the Jenkins “arm” jobs so as to differentiate 32bit / 64bit architectures
- Full Pipeline transformation for EdgeX services
## EdgeX DevOps Commitments (Fuji)

### Scope of Work

<table>
<thead>
<tr>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add static artifact analysis into the EdgeX Jenkins Pipeline (analysis of Docker /runtime artifacts, not the source code)</td>
<td></td>
</tr>
<tr>
<td>Add code and artifact signing with semantic versioning</td>
<td></td>
</tr>
<tr>
<td>Conduct build performance optimizations by:</td>
<td></td>
</tr>
<tr>
<td>• Adding Pipelines for EdgeX Foundry base build images</td>
<td>✔️</td>
</tr>
<tr>
<td>• Allow base build images to be managed locally within Nexus</td>
<td></td>
</tr>
<tr>
<td>• Leverage PyPi Proxy for local pip dependencies</td>
<td></td>
</tr>
<tr>
<td>Explore static code analysis like Checkmarx, Coverity, GuardRails, Synk, SonarQube</td>
<td></td>
</tr>
</tbody>
</table>

### Notes:

- **Clair Server landing at Risk for Fuji**
  - Work Around will be to use Intel hosted Clair server until decision is made by LF to support landing dedicated infrastructure
  - Work related to automating the scans as part of the build, will defer to Geneva scope

- **gitsemver along with lftools used for artifact signing and semantic versioning**

- **Jenkins build performance optimizations for base build images completed**

- **All base build images will now be stored in Nexus (Snapshot):10003**

- **PyPi enabled as part of Edinburgh scope**

- **Initial review of GuardRails showed that the product was identifying issues which were not applicable for microservices architecture**
Edinburgh Release

Release Planning
Edinburgh Dates

• Freeze Date – May 28
• Release Date – June 20
• Press Release – July 11
• Dot Release – July 22
## Past / Future Agenda Topics

<table>
<thead>
<tr>
<th>WW27</th>
<th>No Meeting – US Holiday</th>
</tr>
</thead>
<tbody>
<tr>
<td>WW28</td>
<td></td>
</tr>
<tr>
<td>WW29</td>
<td></td>
</tr>
<tr>
<td>WW30</td>
<td></td>
</tr>
<tr>
<td>WW31</td>
<td></td>
</tr>
<tr>
<td>WW32</td>
<td></td>
</tr>
<tr>
<td>WW33</td>
<td></td>
</tr>
<tr>
<td>WW34</td>
<td></td>
</tr>
<tr>
<td>WW35</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Athens Project</strong> – proxy server for go package dependencies</td>
</tr>
<tr>
<td></td>
<td><strong>Community Involvement</strong></td>
</tr>
</tbody>
</table>