2023-03-02 Infrastructure Working Group Agenda and Meeting Notes

March 2, 2023

02 Mar 2023

Antitrust Policy Notice

Linux Foundation meetings involve participation by industry competitors, and it is the intention of the Linux Foundation to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of, and not participate in, any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

Examples of types of actions that are prohibited at Linux Foundation meetings and in connection with Linux Foundation activities are described in the Linux Foundation Antitrust Policy available at http://www.linuxfoundation.org/antitrust-policy. If you have questions about these matters, please contact your company counsel, or if you are a member of the Linux Foundation, feel free to contact Andrew Updegrove of the firm of Gesmer Updegrove LLP, which provides legal counsel to the Linux Foundation.

When: Thursday, March 2, 2023
9am to 10am PDT/12:00PM-1PM EST
ZOOM Please Register in advance for this meeting:
https://zoom.us/meeting/register/tJwpcO2spjkpE9a1HXBeyBxz7TM_Dvo8Ne8j

Attendees:

- Sean Bohan (openIDL)
- Jeff Braswell (openIDL)
- Nathan Southern (openIDL)
- Peter Antley (AAIS)
- Adnan Choudhury (Chainyard)
- Yanko Zhelyazkov (Senofi)
- Surya Lanka (Chainyard)
- Ken Sayers (AAIS)
- Tsvetan Georgiev (Senofi)
- Allen Thompson (Hanover)
- Aashish Shrestha (Chainyard)

Agenda Items:

- Testnet Experience, Blockchain Automation Framework and the Decision to Use Fabric Operator (Senofi)
- Next topics
- AOB

Minutes:

- History of openIDL (IBM)

- originally used BAF for network deployment and operations
- BAF used to be hyperledger project, was deprecated, part of Bevel
- BAF was outdated, customizations on top of it
- BAF archived as a project, works differently to Fabric Operator
- Operator released recently, announced at Hyperledger Global Forum, saw potential, decided to investigate
- did POC on how to use Operator last Dec
- Viable solution as replacement to BAF

Challenges

Issue	Severity	Impact
Outdated BAF version. No Kubernetes 1.21+ support	Very High	High TCO
BAF is based on HELM and Ansible	Very High	Maintenance
No action to remove orderer/peer	Very High	Functionality
No action to add or remove anchor peers	Very High	Functionality
Orderers count is not a configurable number	Very High	Functionality
Chaincode deployment as an external service is not supported	Medium	High TCO, Reliability

- AWS currently supports Kub 1.25, tried to deploy all on the latest and currently supported version of Kub
- Things change fast in Kubernetes
- for openIDL - need to be on the latest and greatest Kub, business network needs to be secure and acknowledge latest standards
- BAF used HELM, orchestrated with Ansible
- Troublesome - troubleshooting of network ops - everything based on HELM and scripts run in containers, very little to no trace of issues or errors that may arise in execution of operational scripts - tough to troubleshoot issues related to ops -deployment of the network, deploying a peer or orderer or CA is def a good appriach to use HELM and deploy as an app, but operation is a different thing, handled with complex scripts running in dedicated containers that were short lived and impossible to troubleshoot
- big challenge to und whats going wrong without logs
- Peers can comm with other Peers, could set peer as anchor peer but if you forgot to set it get into trouble, cannot remove peer other than manually and need to create new one
- Ordering service preset to 5 orderers - wasn't configurable number, through scripts deployed had to go and change logic to make it configurable, had to do it
- noticed chaincode deployments, gen speaking if you dont change the way it is done, basically handled by the peer, where new container is created and essentially it is managed by the peer on its own, but they faced an issue where container may get stale, may not get proper response, wanted to have the chaincode in its own pod, deployed by sep process - not directly managed by a peer
- Peer by default uses docker container to manage chaincode, dont see that deployed, dont see it in Kub, like a background process, managed by the peer, a problem when peter restarts those containers and they are out of management of Kub (no control or insight into whats going on)
- approach to handle chaincode deployments in Kub and recommended by the community
- Chaincode deployment issue is fixed in the last couple of months, if you restart peer the chaincode containers still exist, currently running in their environment successfully
- Different builder in peer? not using external service, but docker-in-docker container, chaincode runs inside that
- Still cannot manage as Kub resource? No, cant. Persisitence of chaincode and containers fixed, whenever any peer restarts, can invoke or querry without fail
- Deploy BAF on Kub 1.22 as well, running successfully in their two enviornments
- Ordering system deployed thru system channel - this will be a deprecated approach - needs headstart on that by removing creation and management of system channel and using system-less approach for ordering service (any org any time) and addresses scalability of ordering service in better way, very imp thing had to consider moving forward

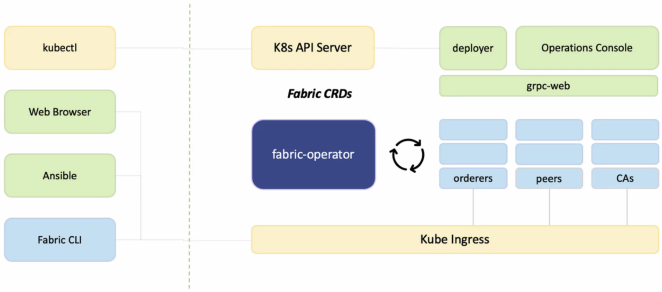
Challenges (continued)

Issue	Severity	Impact
Multiple layers of invocation to perform an operation	High	High TCO
Anchor peer can be added only when adding new org to the channel or when creating new channel	High	High TCO, Functionality
There is no dedicated TLS CA server to issue and manage the record of TLS certs	High	Security

- 3-4 levels of nesting made it hard to understand what was going on and troubleshoot
- thought of that as high TCO issue, hard to fix problems

- Anchor peers are same problem
- no dedicated TLS CA server, not good practice, important security issue too
- chose the ones related to operator and the basic reasoning

Fabric operator architecture



SENOFI

- Operator is native implementation on top of Kub, written in GO (GOLang), follows pattern of any Kub resource
- Operator logic is inside Kub
- if peer needs an identity from the CA, if the operator knows there is no CA will not bring up peer, will wait for certain events to happen, beauty of internal operator logic
- working out eventually
- allows to create identities, do any op on the network in this interface, maybe an issue at 100 nodes but helps with a smaller size network, speeds things up, ease of admin
- Ansible collections - opensourced, work with the Operator inside of Kub - use interface to implement Operator and have front end which can help with network setup and ops
- fabric operator console - manage on the same deployment, cluster, multiple orgs
- makes it possible to use same Kub clusters for dev net to spawn network of mult orgs without deploying on mult AWS accounts
- optimizes costs of dev networks needed to stage dev in the future

Fabric operator advantages

- Native Kubernetes implementation (Controller based on the Operator pattern)
- Operator console (UI)
- Hides complexity and avoids complicated flows
- Low integration cost

SENOFI

- lower cost than implementing Bevel
- Operator addresses functional and security gaps, adding full capabolity of managing fabric network, not just related to new peers or chaincodes, addresses cert rotations when certs expire in the future
- testnet certs expire in a year, Oct/Nov 2023 and the only way to rotate is to do it manually, highly complex task using Fabric Native CLI, with controller much easier way to do it
- governance, access control, creating resources, read-write - all possble with Operator, harder to say with Bevel
- Easier to do things from UI than from script
- What does operator NOT do?
    - doesn't do applications, those are still something that are not in scope of operator
    - setting up or deploying apps, out of scope for operator
    - solely for the management and operation of network
    - agnostic, dependency on Kub, should deploy and work on any cloud once managed to deploy (advantage)
    - in openIDL, things tightly coupled with AWS cloud deployment, with operator most will be solved, up to point
    - after you have operator up and running, should work same on any cloud
    - opens door for openIDL to in near future integrate with other cloud providers like Azure

What's missing

- Integration with Vault
- Declarative way to perform operations such as deploy channel and chaincode
- Audit trail is limited as now Flux is not used

SENOFI

- good not perfect solution, certain drawbacks
- major one - no mechanism to keep certs in external secure location (vault, HSM)
- there is HSM integration, not well documented, discovered integration, anything else is not in place, all the admin certs no direct integration with other storage
- where flux was used to update network, could go to git repo and figure out what was changed, now that is more limited, stil have audit trail, going to be in the ansible or AWX used to run ansible, not diminshed but changed place from git repo to ansible execution log
- AWK a server to run ansible scripts, create projects, jobs - have ability to maange mult ansible scripts
- used in more uniform way, now all the logs exist in ansible
- in Kub, limited to what Kub logs, can take on task to figure out implement something with better audit trail
- AC - mentioned used Operator for Testnet, used it to get network up and running?
    - current testnet deployed with existing openIDL gitops tool, managed to deploy network on separate dedicated AWS instance with operator including applications
- JB - plan is, once it is proven will be platform for testnet - currently in parallel mode
- config tightly coupled with repo in gitops
- fork to config and do things there, could go with dedicated private repo
- idea behind chang e- dont want people to fork repo, it is immutable and read-only so every project should be using openIDL git repo to run the network and only contribute fixes to the code while other configs run outside
- example: another POC in the future, goal is to in the POC the only custom thing you may have other than apps but from network perspecitve is dedicated network in private repo without forking openIDL codebase

Action items:

- [ ]
- [ ]
- [ ]