Jakarta Architecture Committee Policy Framework

2022-01-25

General Policy Framework Improvements 1/2

- Monitoring Enhancements:
  - Metrics extensions and Prometheus support (General)
  - Improve monitoring capabilities in policy components (send app metrics to Prometheus)
- Actor Model extensions: Driven by internal/external needs (General)
- Recovery Enhancements: Detection and recovery of misbehaving control loop applications (PDP-D)
- Multi-cluster deployment investigations (General)
- Allow underlying database to be configured
  - Default continues to be MariaDB
  - other databases such as PostgreSQL will be possible to configure

General Policy Framework Improvements 2/2

- Improve robustness of the Policy Framework OOM deployment by improving readiness probes
- Replacement of some Policy Framework utility libraries with Spring Framework support
  - Replace Lifecycle Management and REST implementation
    - CLAMP/Control Loop and PAP already done
    - Other components to be done in Jakarta and later releases
  - Introduce Spring Framework transactions
- Improvements in XACML-PDP

Handling of Policy Type Metadata

- Rule sets and other metadata for policy types are preloaded, pre-configured, packaged at design time, or passed in all policies
- This improvement allows rule sets and other metadata for policy types to be set at run time
- Pass through mechanism for passing PDP-specific information
- Makes it easier to use the Policy GUI to parameterize policies for all PDPs
- See the Policy Framework Wiki for more information

Database Related Items

- Ability to group DB changes into a single transaction
- Use of a connection pool instead of single connections
- Move from dao-pattern to repository pattern
- Allow databases other than MariaDB to be used
- Backup and Restore

Use of TOSCA Control Loop approach for all Control Loops

- CLAMP flow for Control Loop Deployment using Cloudify is broken, Control Loop deployment in DCAE using Cloudify blueprints no longer works
- The recommended approach is to use the TOSCA based approach and retire support for the Cloudy approach
- For the TCA use case, a study on its implementation using the TOSCA/Participant approach is ongoing
- The CLAMP front end will remain, as it supports the TOSCA/Participant approach and also supports Policy creation and Service Template
- The DCAE design for blueprint processing in DCAE-MOD now works towards a chart museum, so this part of the flow will change
- Current workflows may be replaced with manual steps and/or with TOSCA/Participant Commissioning/Instantiation flows
- SDC is being updated in Jakarta to support TOSCA based control loops
- We will have to change pairwise testing and tests run by OOM to use the changed Control Loop implementation
- Other Control Loops that use a Cloudify approach will be impacted

TOSCA Control Loop Improvements

- Support for Control Loop design
  - Onboarding of Control Loop Elements in SDC
  - Composition of Control Loops in SDC
  - Distribution and commissioning of Control Loops using the SDC
- Server Improvements
  - Spring Framework for Persistence
  - Better support for control loop updating
  - Statistics using Prometheus

Other Miscellaneous Improvements

- Policy GUI
  - Merge of the current three clients into a single container
  - Code coverage (policy-gui is on 67%, all other PF repos are on 80%)
- Test Documentation
  - Improvements started in Istanbul will continue
  - Target is to have all CSITs, smoke tests, and Pairwise tests documented
- Contract Testing
  - Contracts might be a useful way of proving the integrity of our APIs
  - Proposed investigation to see how we can marry Swagger and Contract Testing
- Release Process
  - Interim release of the Policy Framework was made in December
  - Scripts developed to aid the release process
  - See the separate Interim Release Session

Changes Impacting the Policy Framework

- Logging
- Service Mesh
- Asynchronous Event Handling (DMaaP, Kafka, Strimzi)
- Others

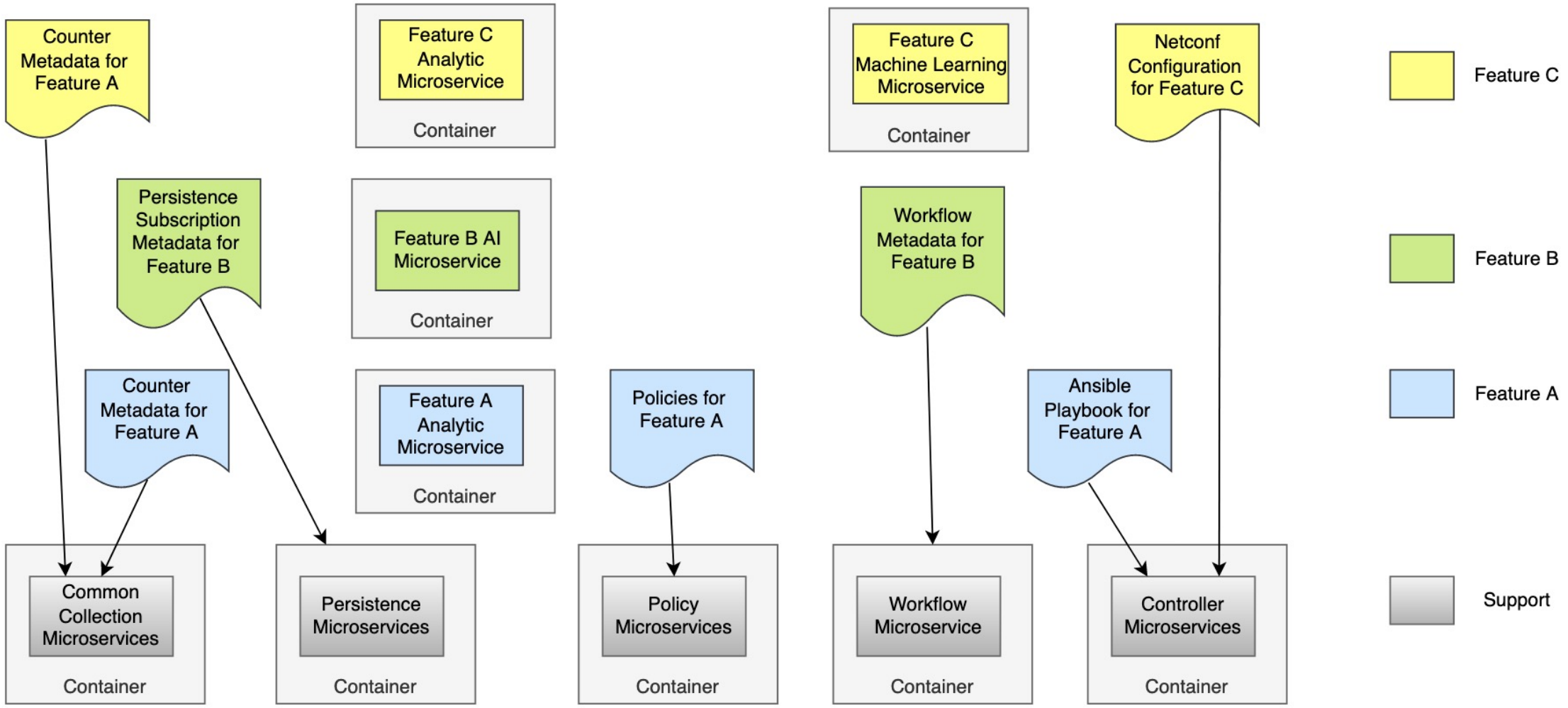
Stretch Items for Jakarta (Miscellaneous)

- Trial a distributed deployment of the Policy Framework
- PoC for persisting states in PDPs
- Study on storing TOSCA models by mapping them directly to one document
  - We can use same Java model we have already defined
  - Less number of access to database thus improving performance
  - More maintainable
  - Can use Spring Framework persistence mechanisms directly
- High Availability
  - Investigation of database high-availability options
  - Replication & Synchronization
  - Failover
  - Load Balancing & Proxy Considerations

Stretch Items for Jakarta (Control Loop)

- Possible improvements in Jakarta and beyond
  - Possibility to list available microservices and other control loop elements at design time
  - Possibility to list running/viable microservices and other control loop elements at run time
- Proposal: Automation Composition Management
  - Proposed PoC in Jakarta
  - Generalization of the concept to include use cases beyond control loops
  - Use cases with arbitrary components working together to deliver a feature such as open loops or collections of features

Automation Composition Management




ONAP
OPEN NETWORK AUTOMATION PLATFORM