OWASP DEPENDENCY-TRACK

Community Meeting
May 2025

Organizational

- Community meetings are recorded and uploaded to YouTube
- Slides will be published in the `DependencyTrack/community` repository
- Please use the Zoom chat to ask questions during the presentation
- There will be an open Q&A section towards the end

Agenda

1. Overview: v5
2. Functional changes
   a. Policy
   b. Access Control
3. Non-functional changes
   a. Database
   b. Async Processing
   c. Data Retention
4. Notes for Contributors
5. What’s next

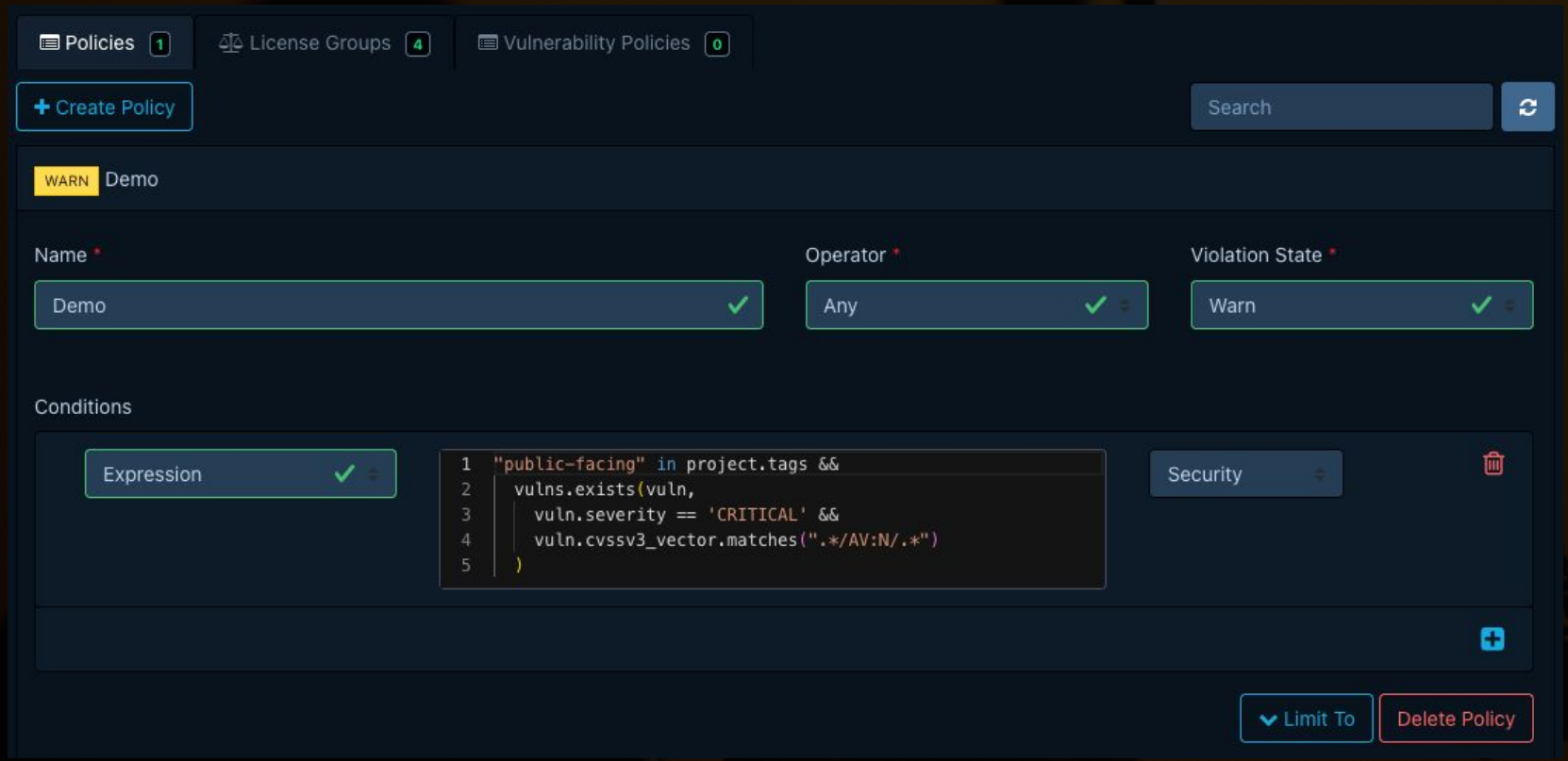
Overview: v5

- Codenamed hyades
- Objectives:
  - Enable Dependency-Track to handle portfolios spanning hundreds of thousands of projects
  - Improve resilience of Dependency-Track, providing more confidence when relying on it in critical workflows
  - Improve deployment and configuration management experience for containerized / cloud native tech stacks
  - Become platform for more supply-chain related risk tracking
- https://github.com/DependencyTrack/hyades
- https://github.com/DependencyTrack/hyades-apiserver
- https://github.com/DependencyTrack/hyades-frontend

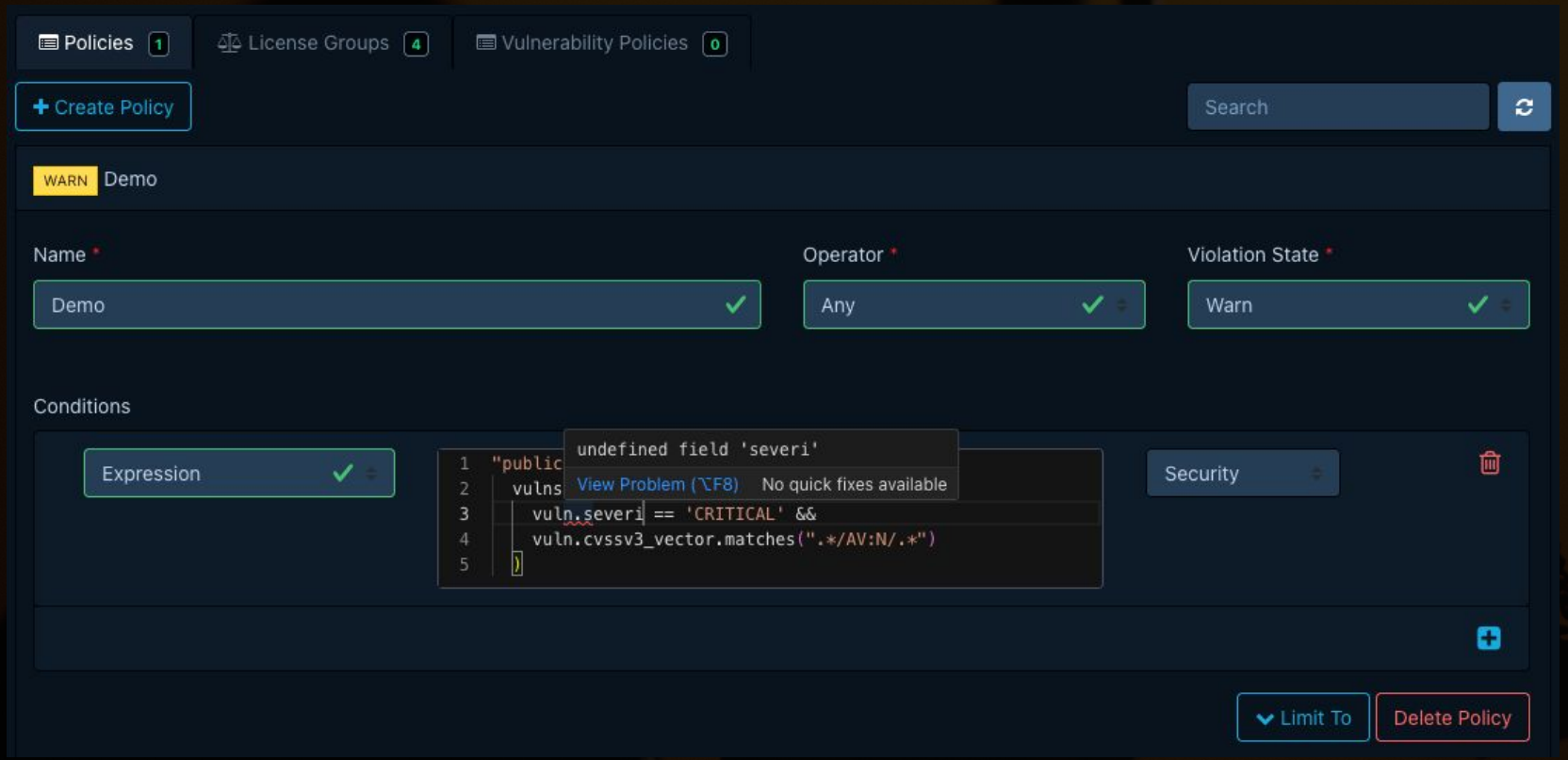
FUNCTIONAL CHANGES


POLICY

Policy: CEL Expressions



Policy: CEL Expressions



Policy: Behind the scenes

- Engine entirely powered by CEL expressions
- “Legacy” conditions transparently converted to CEL
- Inputs defined in Protobuf
  - Type-safe
  - Easy to ensure backward-compatibility (checked in CI with buf)
- Only data required by expressions is loaded from database
- Compared to solutions like OPA, no networking overhead
- CEL is not a scripting language (that’s good)
- Custom functions to provide domain-specific functionality
  - e.g. vers-powered version range checks, dependency graph traversal

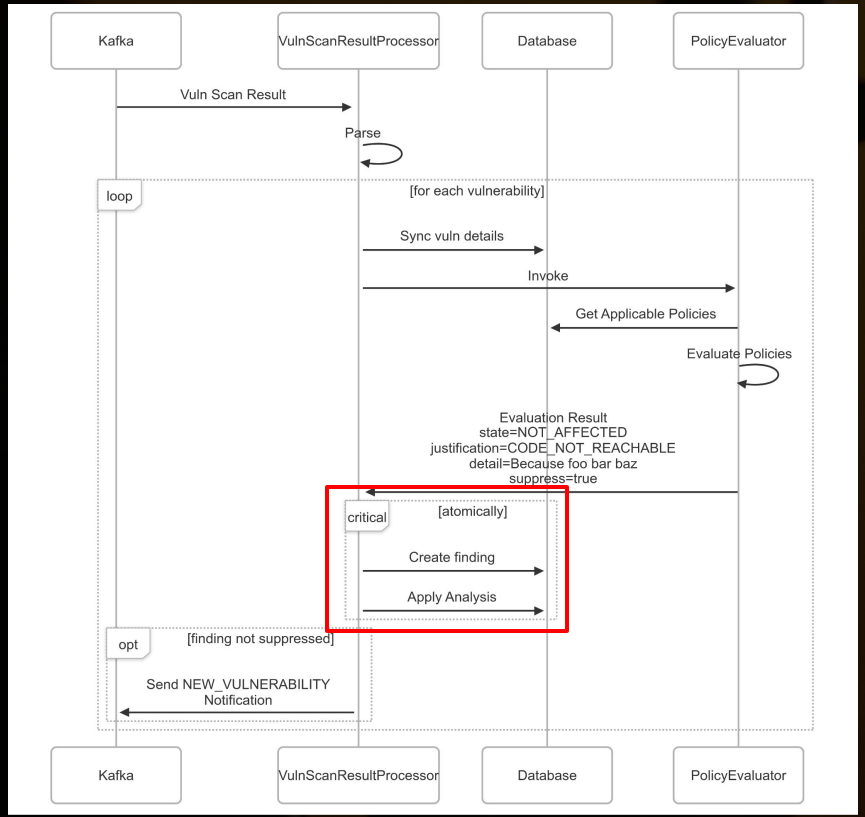
Policy: Automatic Vulnerability Auditing

CEL

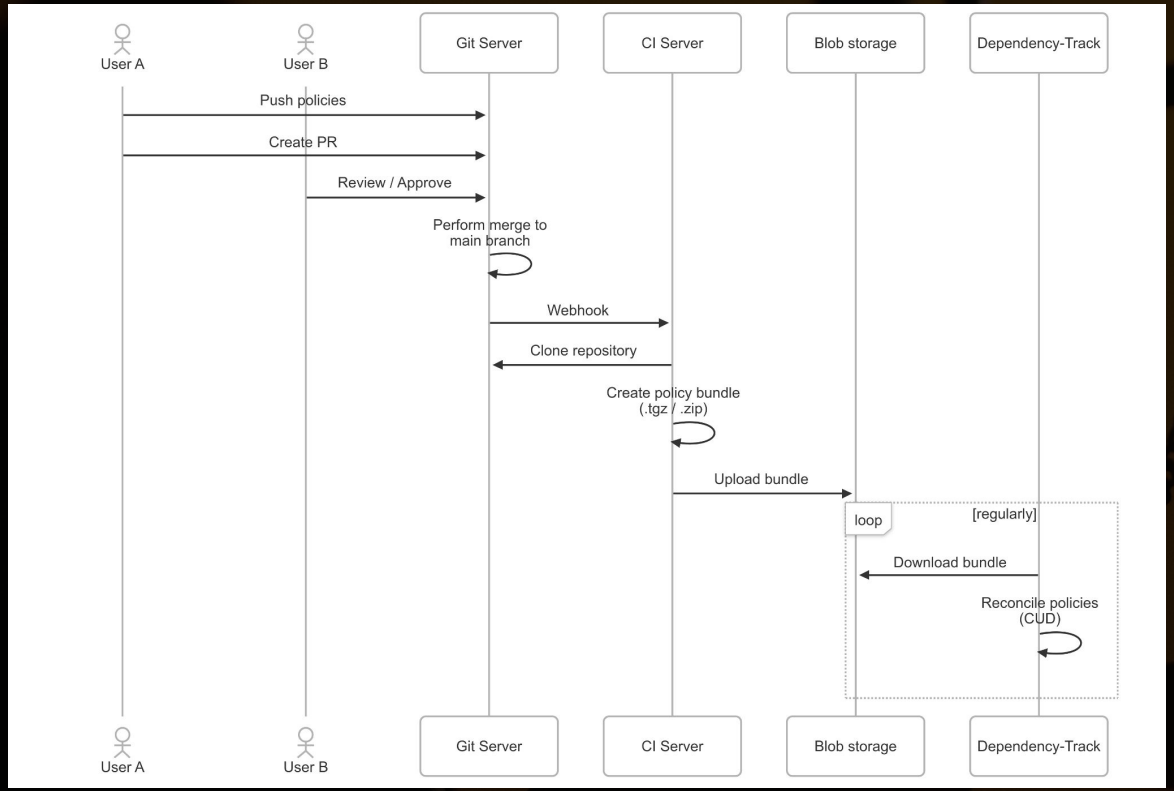
“VEX”

```yaml
apiVersion: v1.0
type: Vulnerability Policy
name: Spring Cloud CVE-2022-41852 Suppression
description: |-
  Suppresses occurrences of CVE-2022-41852 in commons-jxpath,
  when commons-jxpath is introduced through Spring Cloud.
operationMode: APPLY
author: Jane Doe
conditions:
- |-
  vuln.id == "CVE-2022-41852"
    && component.name == "commons-jxpath"
    && component.is_dependency_of(org.dependencytrack.policy.v1.Component{
         group: "org.springframework.cloud",
         version: "vers:maven/>3.1|<3.3"
       })
analysis:
  state: NOT_AFFECTED
  justification: CODE_NOT_REACHABLE
  details: |-
    It was determined that CVE-2022-41852 is not exploitable in Spring Cloud
    components, because the vulnerable code is not used.
  suppress: true
```

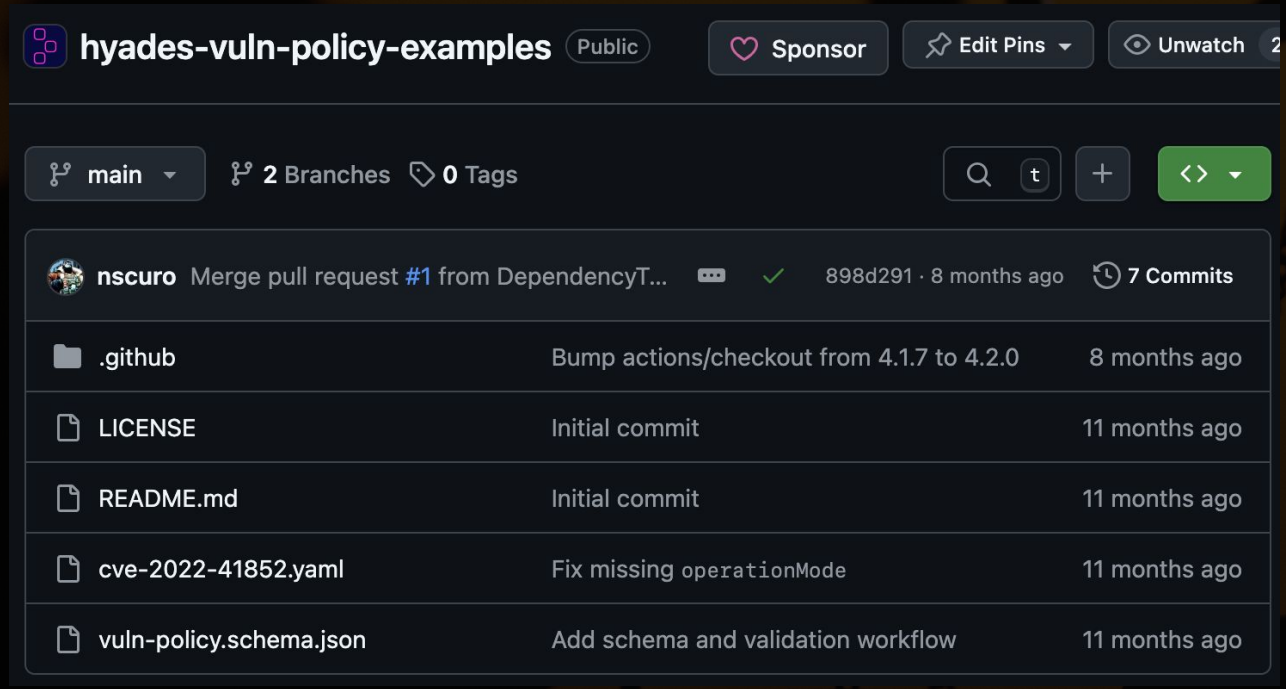
Vulnerability Policy: Evaluation



Vulnerability Policy: Lifecycle

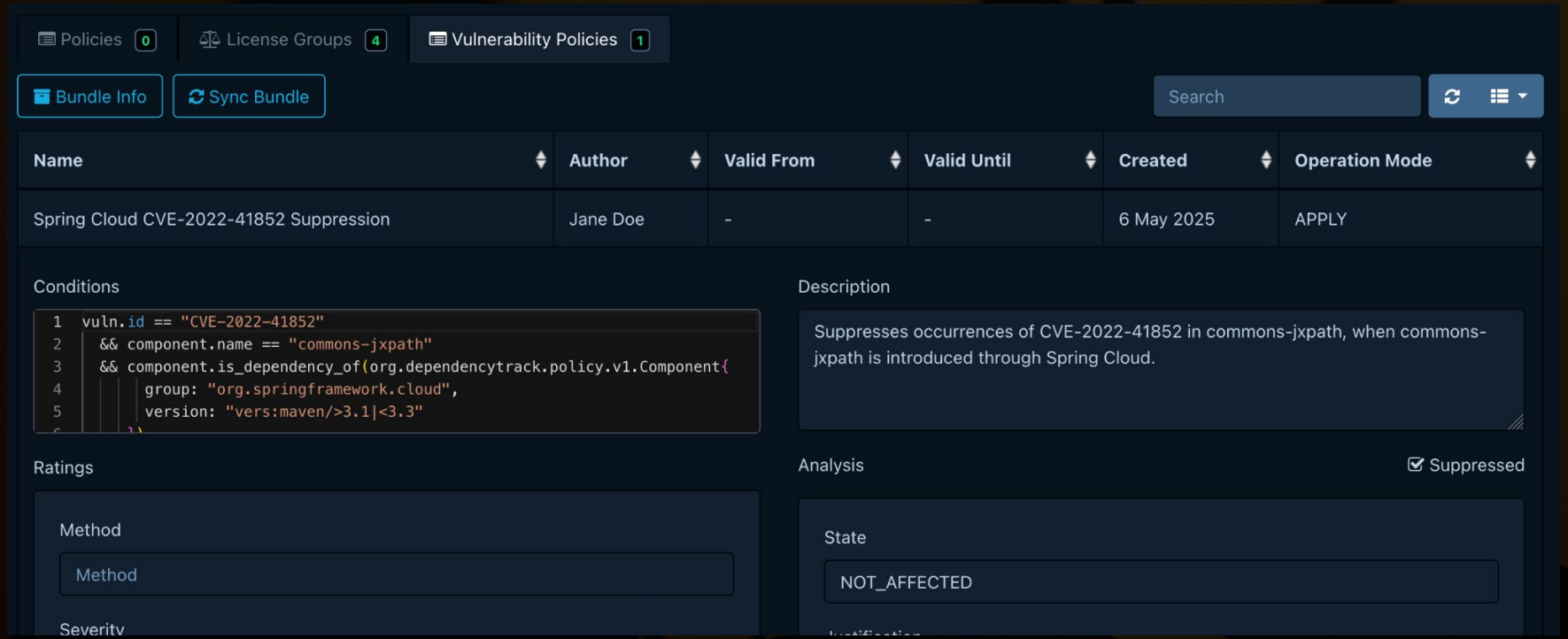


Vulnerability Policy: VCS Management

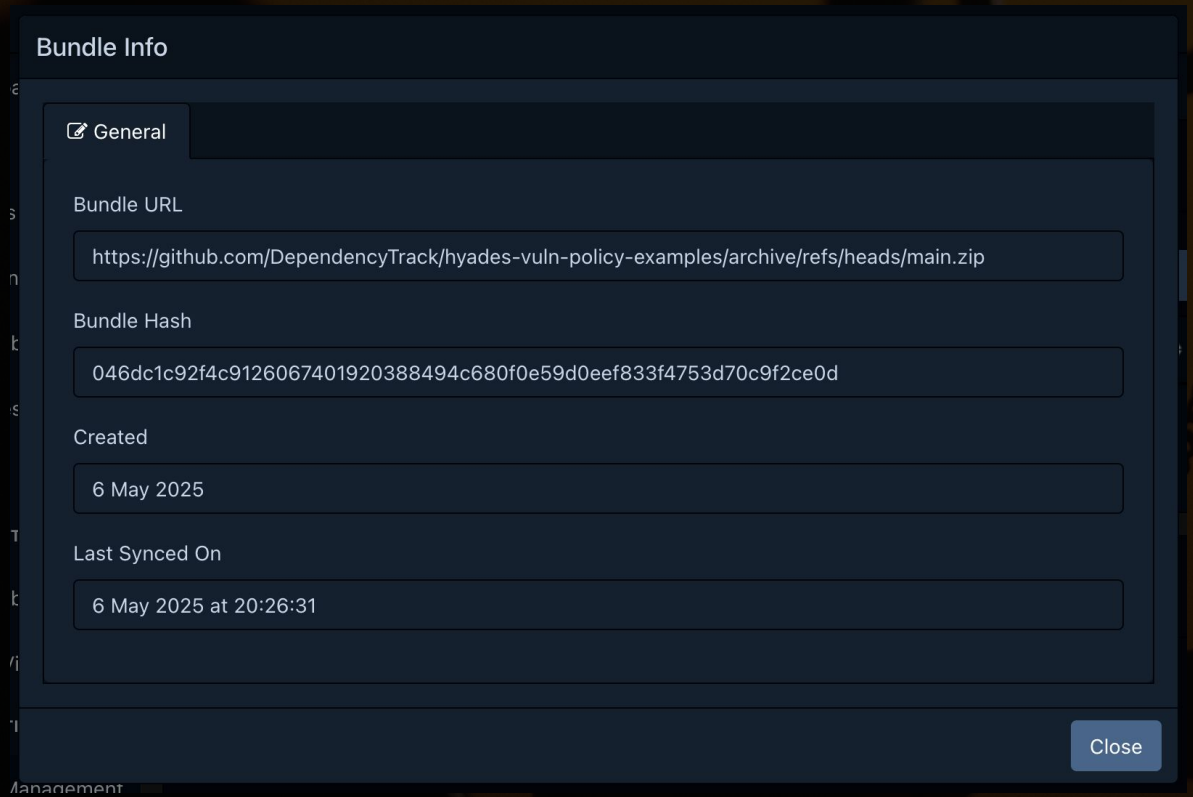


https://github.com/DependencyTrack/hyades-vuln-policy-examples

Vulnerability Policy: UI



Vulnerability Policy: UI

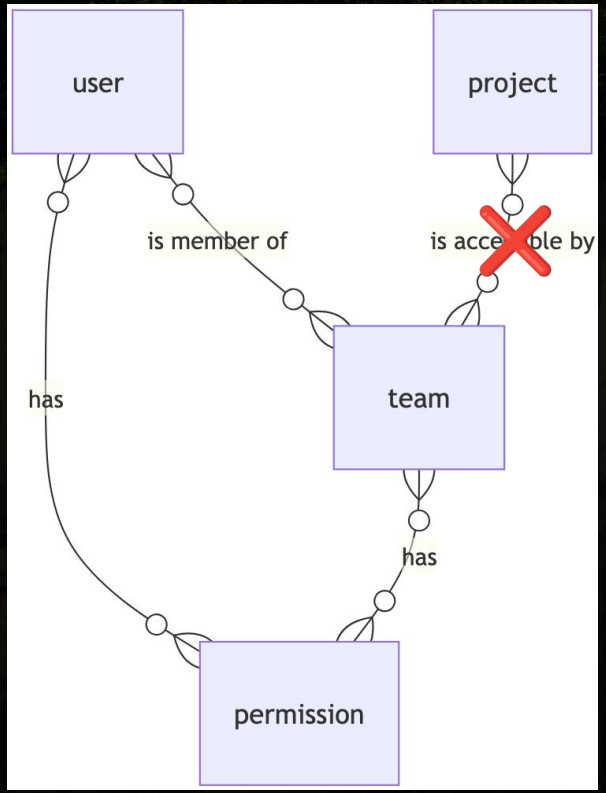


Policy: To Be Done

- CEL should be first choice for conditions
  - Editor to be the central UI element
  - Legacy conditions offered as “templates”
- Vulnerability policies should be manageable via UI
- Improve UX for authoring expressions, editor is still a bit janky
- Offer autocomplete


ACCESS
CONTROL

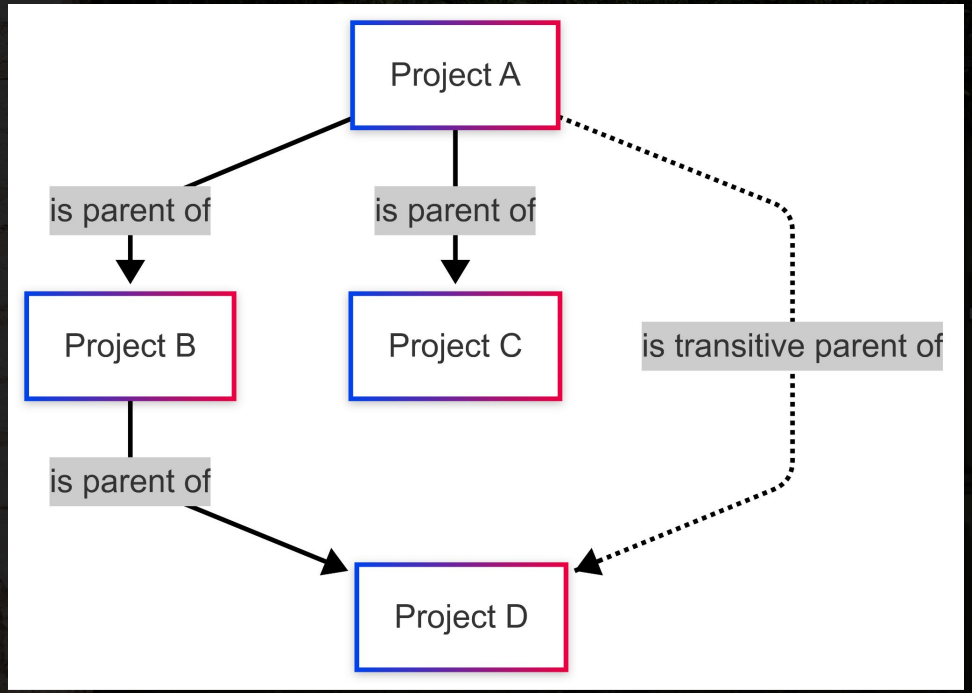
Portfolio Access Control: Refresher



Portfolio Access Control: Closing Gaps

- Feature still in preview in v4, with known gaps
- Goal: production-readiness by v5
- Most known gaps already closed
  - TODO: Filtering of portfolio metrics based on project access
- Inheritance now supported

Portfolio Access Control: Inheritance



Portfolio Access Control: Status

hyades/#1645

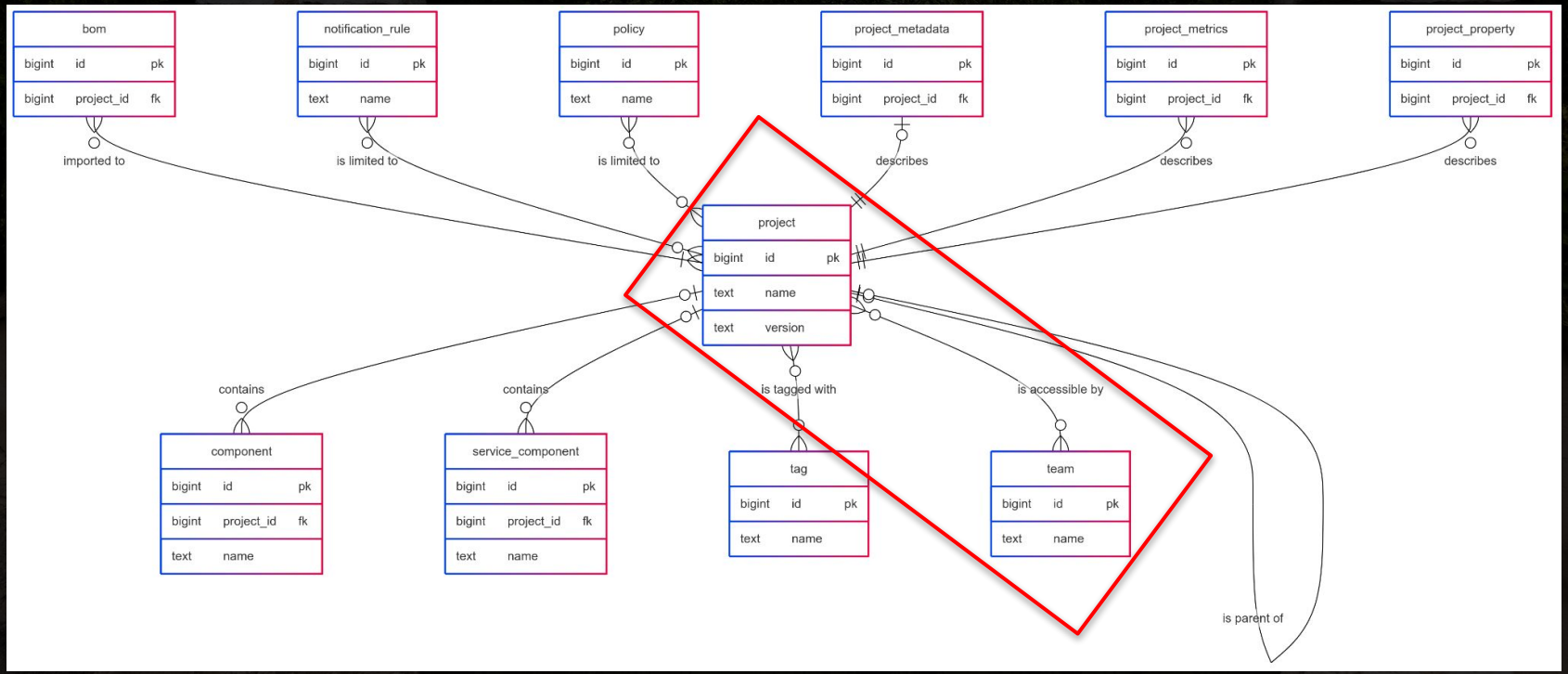
Bring the "Portfolio Access Control" feature into a final, production-ready state

https://github.com/DependencyTrack/hyades/issues/1645

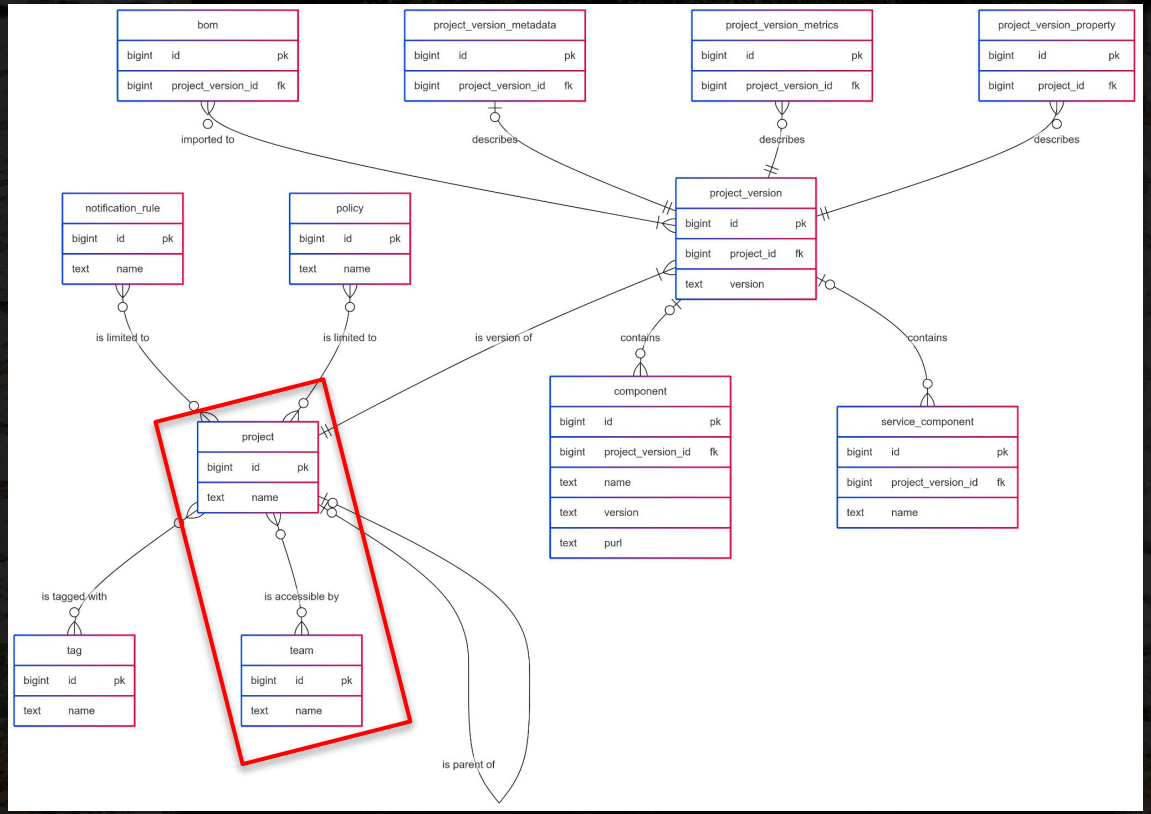
Portfolio Access Control: Granularity

- Current model:
  - Coarse permissions, e.g. `PORTFOLIO_MANAGEMENT`
  - Global permissions that apply to all accessible projects
  - Teams must be granted access to individual projects
- Vision:
  - Permissions should be more fine-grained
  - It should be possible to scope permissions to specific projects
  - Project access should be assignable on a per-user basis
  - Project access should apply to all versions of a project
- Challenge: Making it perform well
  - Enforcing for individual projects is easy
  - Bulk operations like “list all accessible projects” is not

Project Model



Project Model: Revamped?



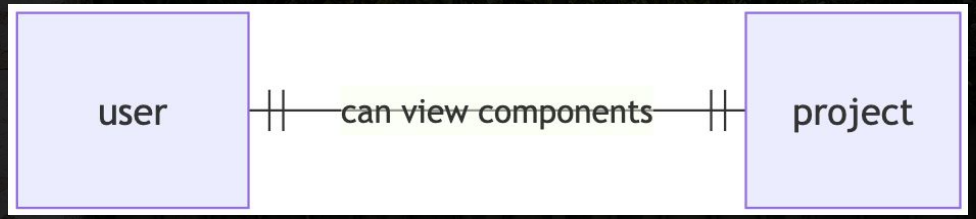
Project Model: Revamped?

hyades/#1736

Decouple the concept of project from project versions

https://github.com/DependencyTrack/hyades/issues/1736

Project-Level Permissions



Project-Level Permissions

hyades/#1632

Project-level RBAC

https://github.com/DependencyTrack/hyades/issues/1632

NON-FUNCTIONAL CHANGES

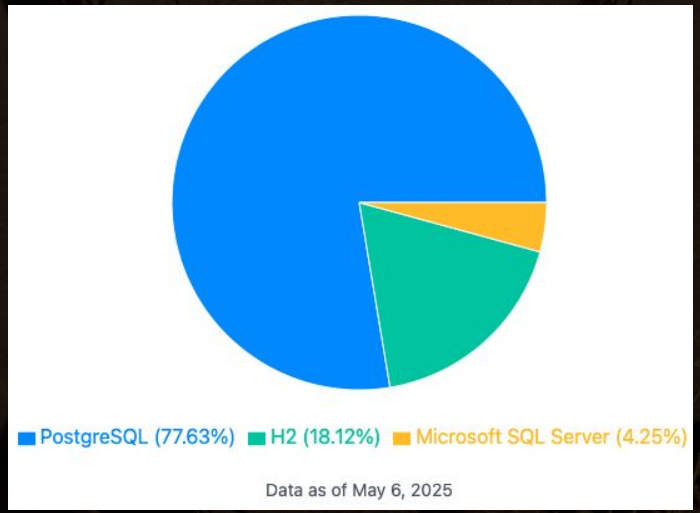

DATABASE

Database

- All-in on Postgres
- Lowest supported version as of now is **13**
  - Might raise to 14 before GA
- Support for H2, MySQL, and MSSQL dropped
  - Drastic reduction in testing effort
  - More opportunities for optimization
- More reliable, changelog-based migrations (Liquibase)
- Ability to use separate set of credentials for migrations
- All tests run against lowest supported version
- Various schema inconsistencies resolved

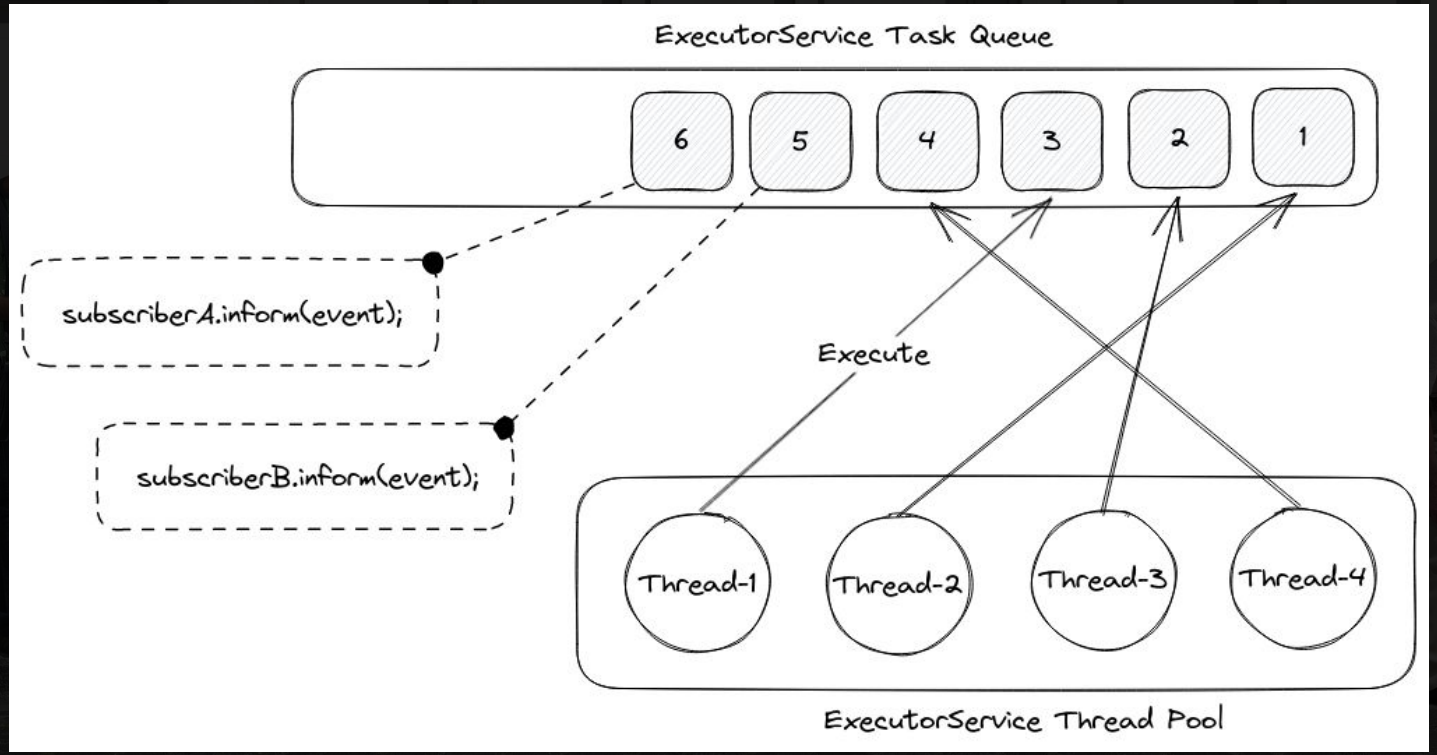
Database: Migration

- Tooling for v4 -> v5 Upgrade will be provided
- When deploying new v4 instances, choosing Postgres is advisable
- Majority of v4.13+ users already use Postgres

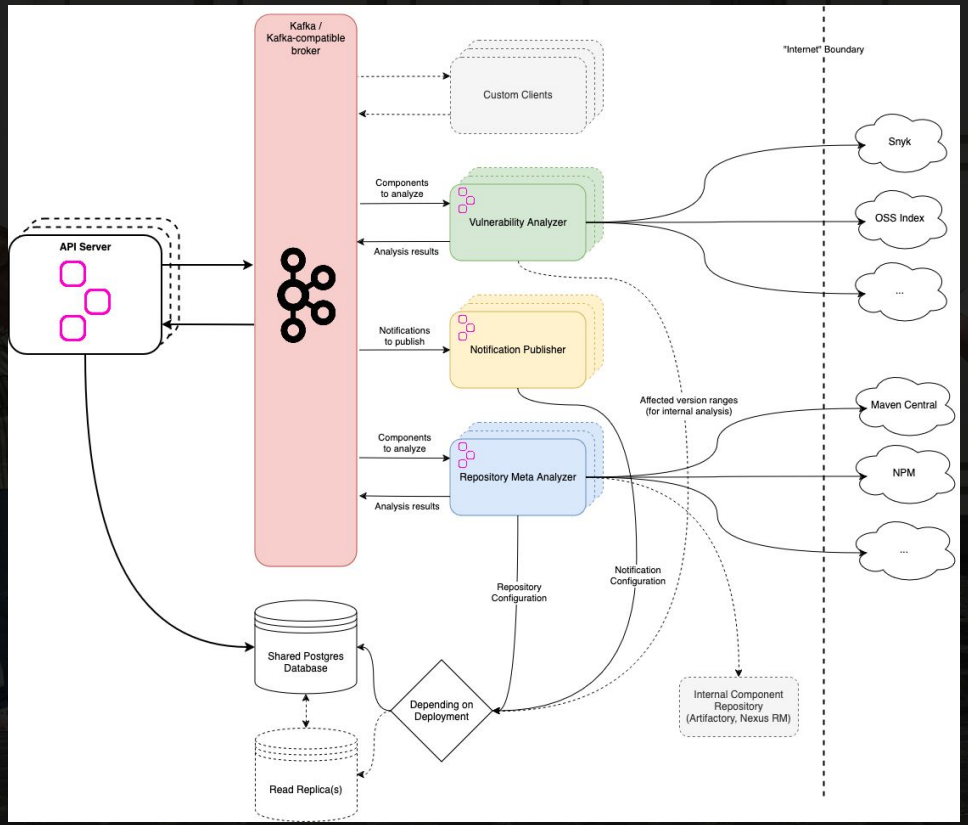



ASYNC PROCESSING

Async Processing: Starting Point



Async Processing: Kafka as Backbone

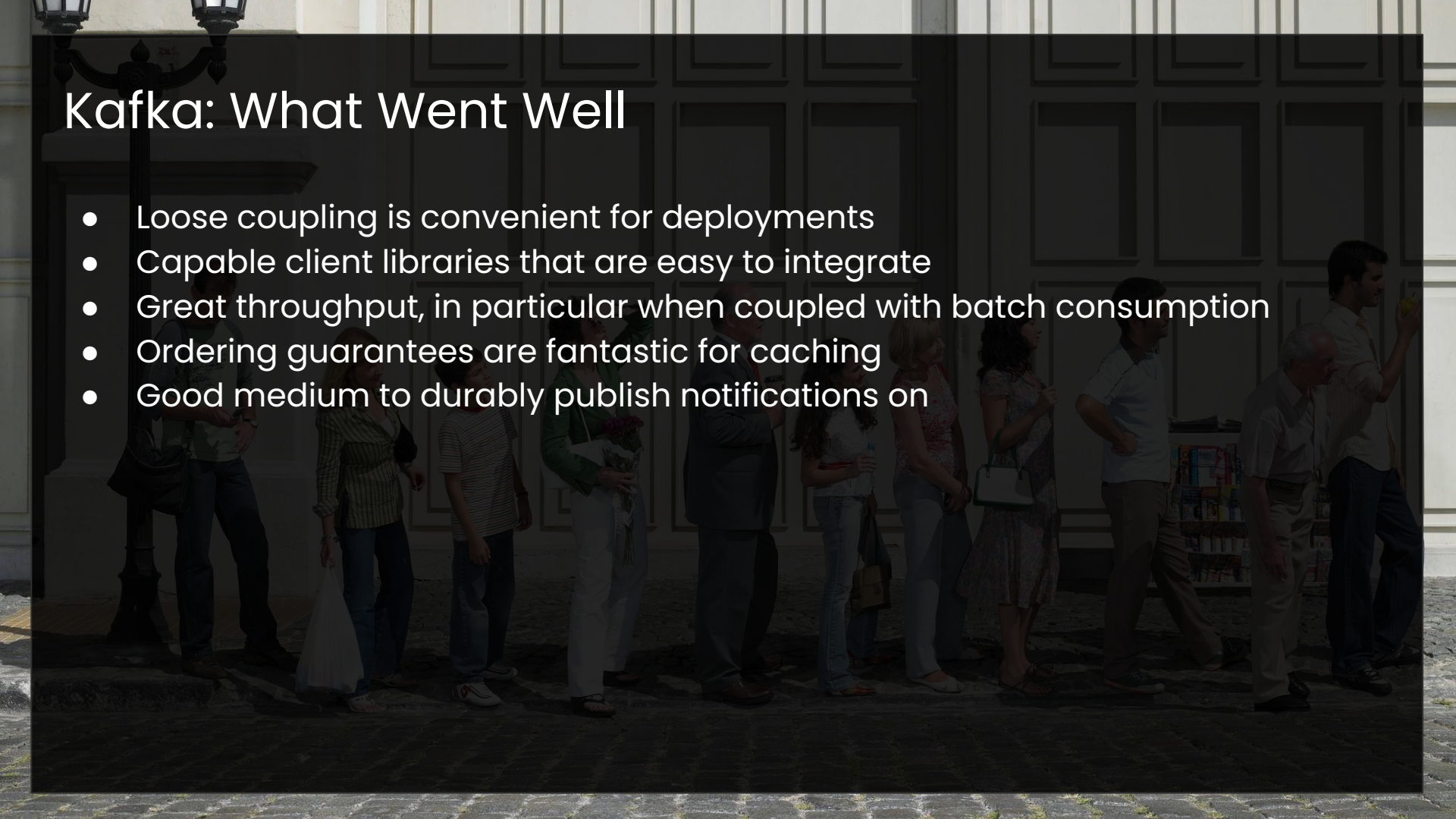


Kafka: Original Intent

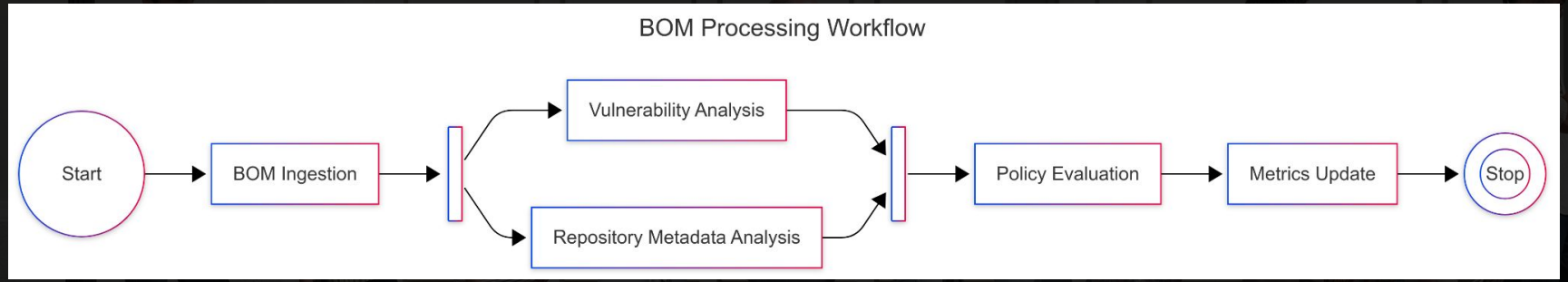
- “Event-Driven”
- Reduce load on API server
- Offload async processing to other services
- Leverage stream processing techniques to reduce DB load
- Encourage loose coupling
- Enable horizontal scaling of services
- Ordering guarantees for lock-free concurrency control
- Tap into large, well-supported ecosystem
    - (Hosting Kafka has become a lot more approachable)

Kafka: What Went Well

- Loose coupling is convenient for deployments
- Capable client libraries that are easy to integrate
- Great throughput, in particular when coupled with batch consumption
- Ordering guarantees are fantastic for caching
- Good medium to durably publish notifications on

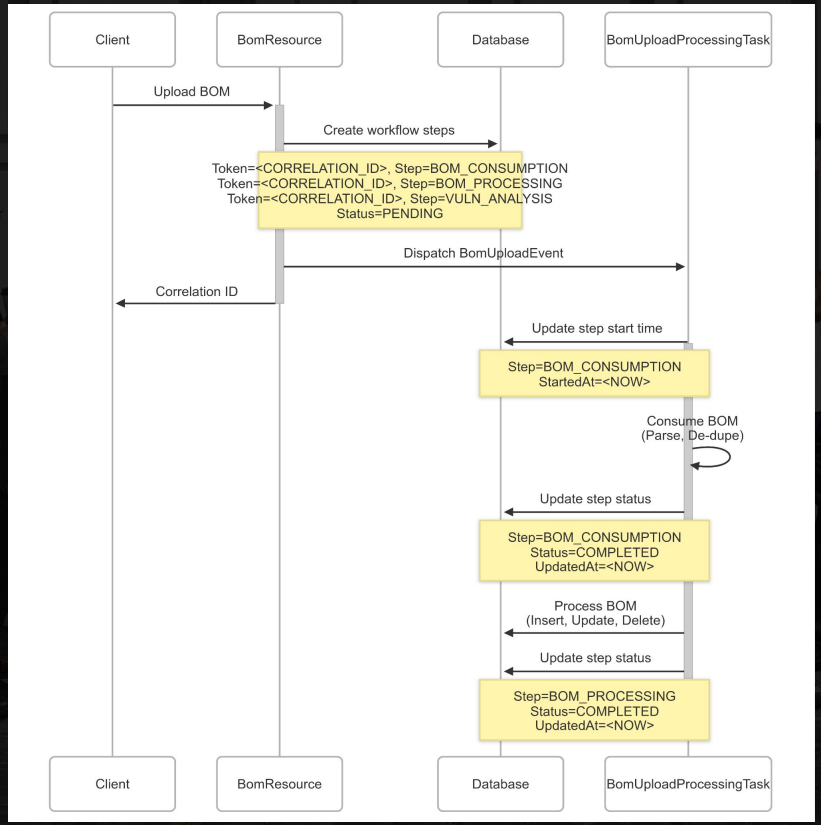


But It Is Workflows, Actually



Tracking Workflow State

Name	Type	Nullable	Example
ID	`SERIAL`	❌	1
PARENT_STEP_ID	`SERIAL FK`	✅	0
TOKEN	`VARCHAR(36)`	❌	`484d9eaa-7ea4-4476-97d6-f36327b5a626`
STARTED_AT	`TIMESTAMP`	✅	`1999-01-08 04:05:06`
UPDATED_AT	`TIMESTAMP`	✅	`1999-01-08 04:05:06`
STEP	`VARCHAR(64)`	❌	`METRICS_UPDATE`
STATUS	`VARCHAR(64)`	❌	`PENDING`
FAILURE_REASON	`TEXT`	✅	`Failed to acquire database connection`

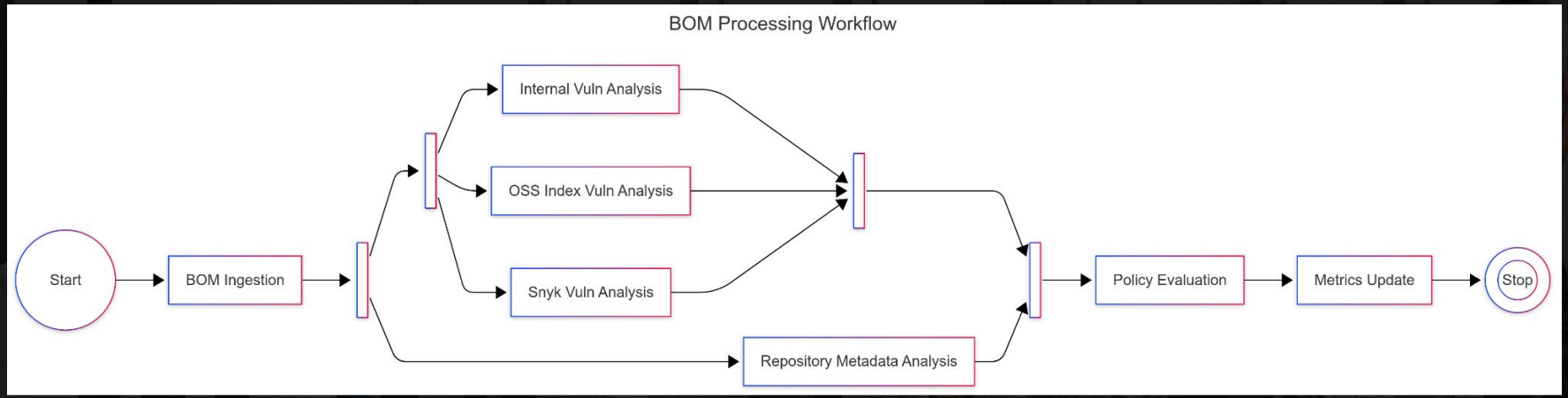


Tracking Workflow State

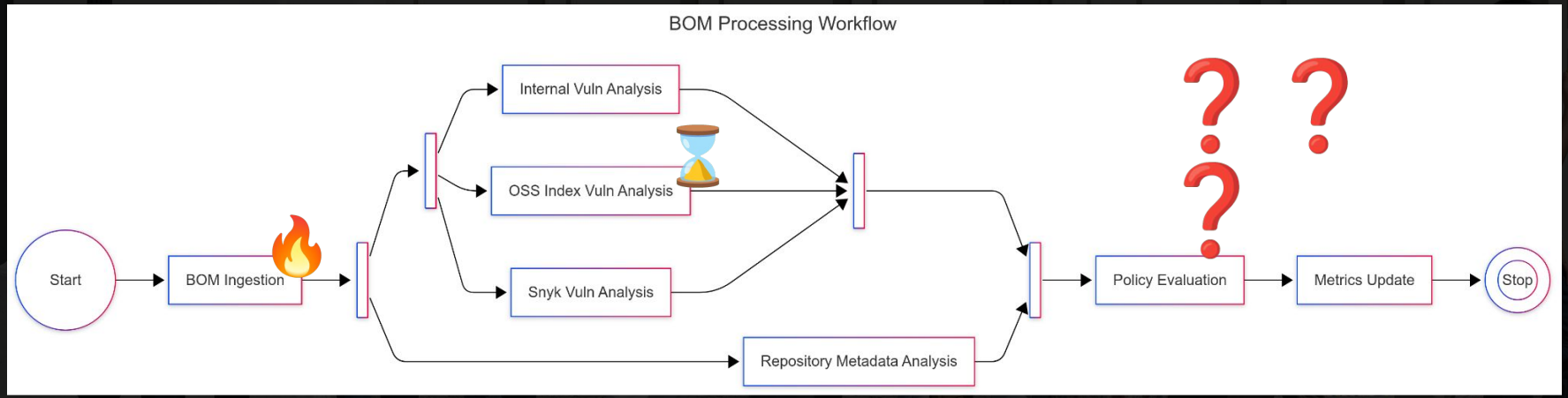
```
GET /api/v1/workflow/token/<CORRELATION_ID>/status
```


```
[
  {
    "step": "BOM_CONSUMPTION",
    "status": "COMPLETED",
    "startedAt": "1999-01-08 04:05:06",
    "updatedAt": "1999-01-08 04:05:06"
  },
  {
    "step": "BOM_PROCESSING",
    "status": "FAILED",
    "startedAt": "1999-01-08 04:05:06",
    "updatedAt": "1999-01-08 04:05:06",
    "failureReason": "Failed to acquire database connection"
  },
  {
    "step": "VULN_ANALYSIS",
    "status": "CANCELLED"
  }
]
```

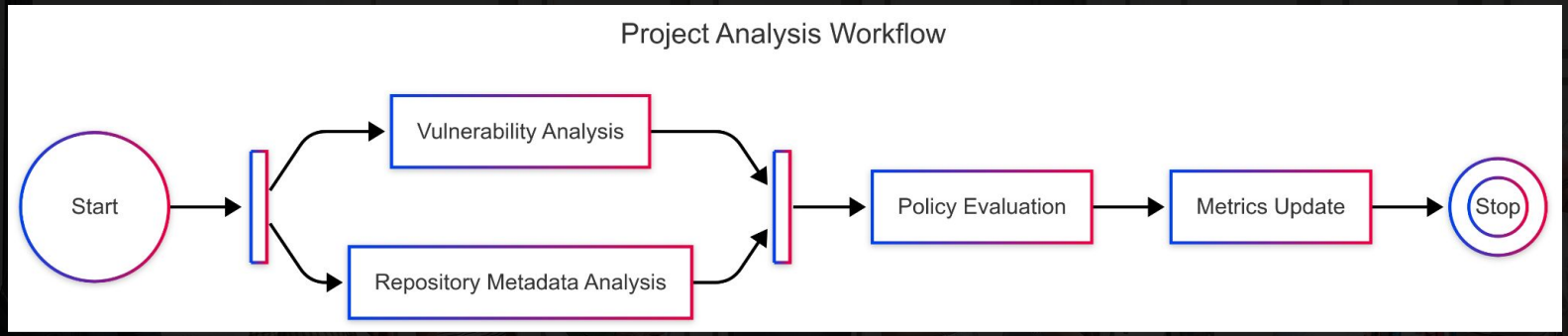
But It Is Complex Workflows, Actually

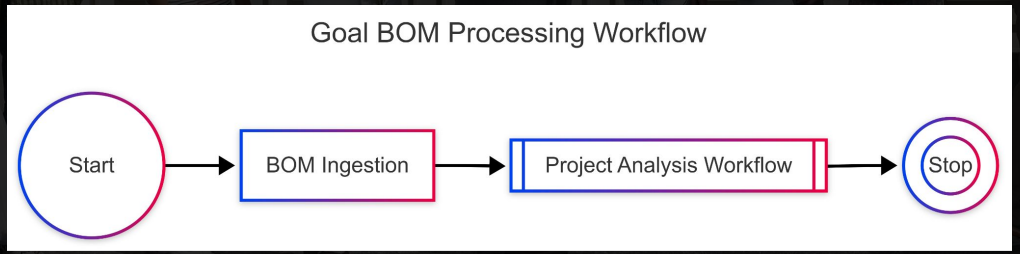


But It Is Complex Resilient Workflows, Actually



But It Is Complex Resilient Reusable Workflows, Actually





Kafka: Why It Wasn’t The Right Choice

- Dual writes threaten reliability of notifications
- Non-blocking retries are hard to achieve
- No support for prioritization
- Message size limitations
- Forces small messages that are processed quickly
- Observability requires special tooling
- End-to-end processes hard to reason about
- Constraints of user environments
- Spotty support for advanced Kafka features like transactions
- Topic management is kind of a mess
- Community support will be hard

Kafka: Letting Go

ADR-001

Drop Kafka Dependency

https://github.com/DependencyTrack/hyades/blob/bc5ac65943f63a6af84eb35d92fc13a62c10bc27/docs/architecture/decisions/001-drop-kafka-dependency.md

What We're Doing Instead

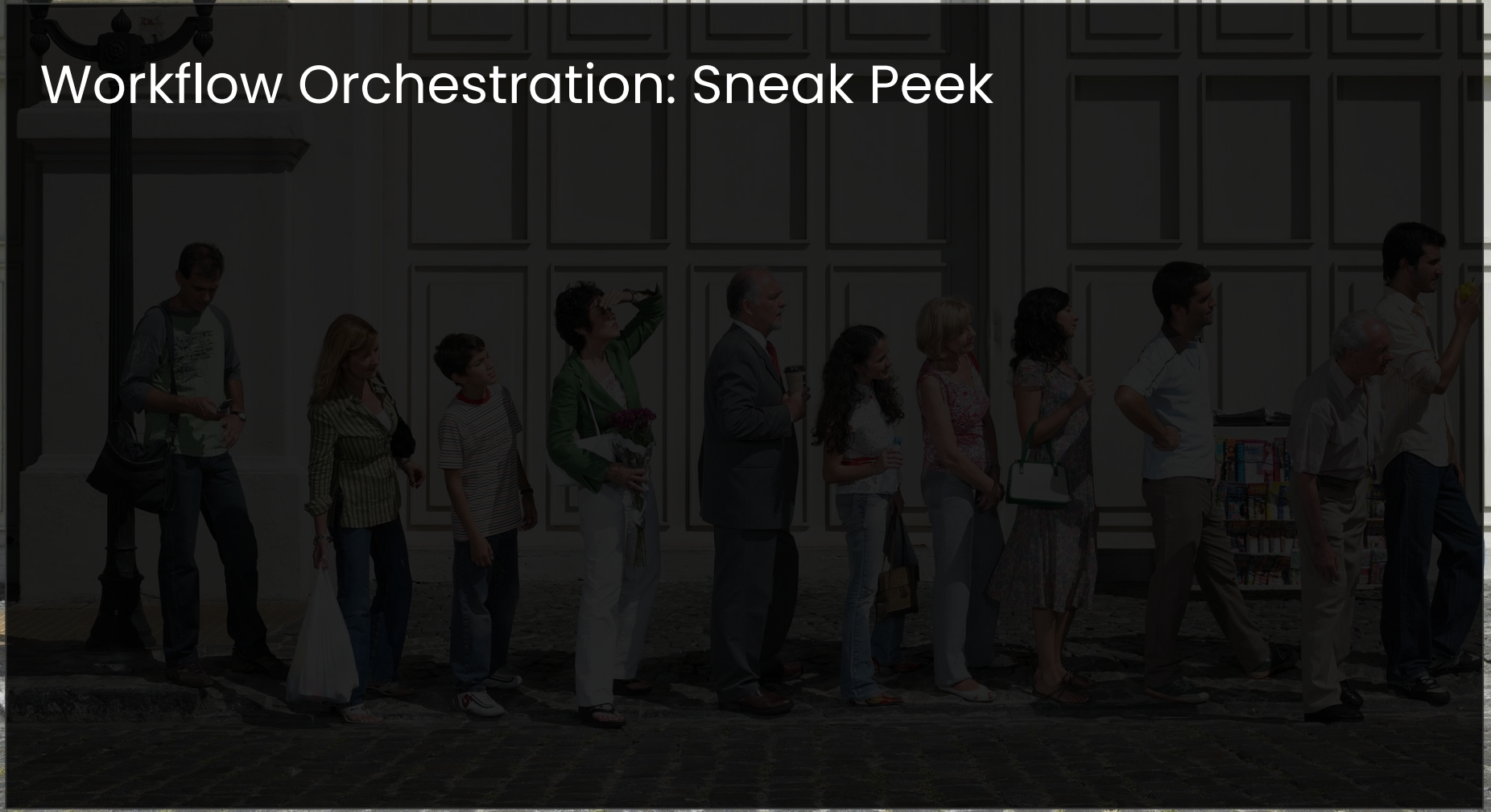
- Choreography -> Orchestration
- Leverage "Durable Execution" model
  - Orchestrations are normal application code
- Heavily inspired by Microsoft's DurableTask Framework
  - Which inspired Cadence, which inspired Temporal
- Embedded engine, no additional services to deploy
- Tasks are executed in lightweight virtual threads
- Message passing via Protobuf
- Optimized for Postgres, optimized for throughput
  - Almost all database interactions use batching
  - Buffering of persistence operations
- Can be configured to use separate database

What We're Doing Instead

ADR-002
Workflow Orchestration

https://github.com/DependencyTrack/hyades/blob/bc5ac65943f63a6af84eb35d92fc13a62c10bc27/docs/architecture/decisions/002-workflow-orchestration.md

Workflow Orchestration: Sneak Peek



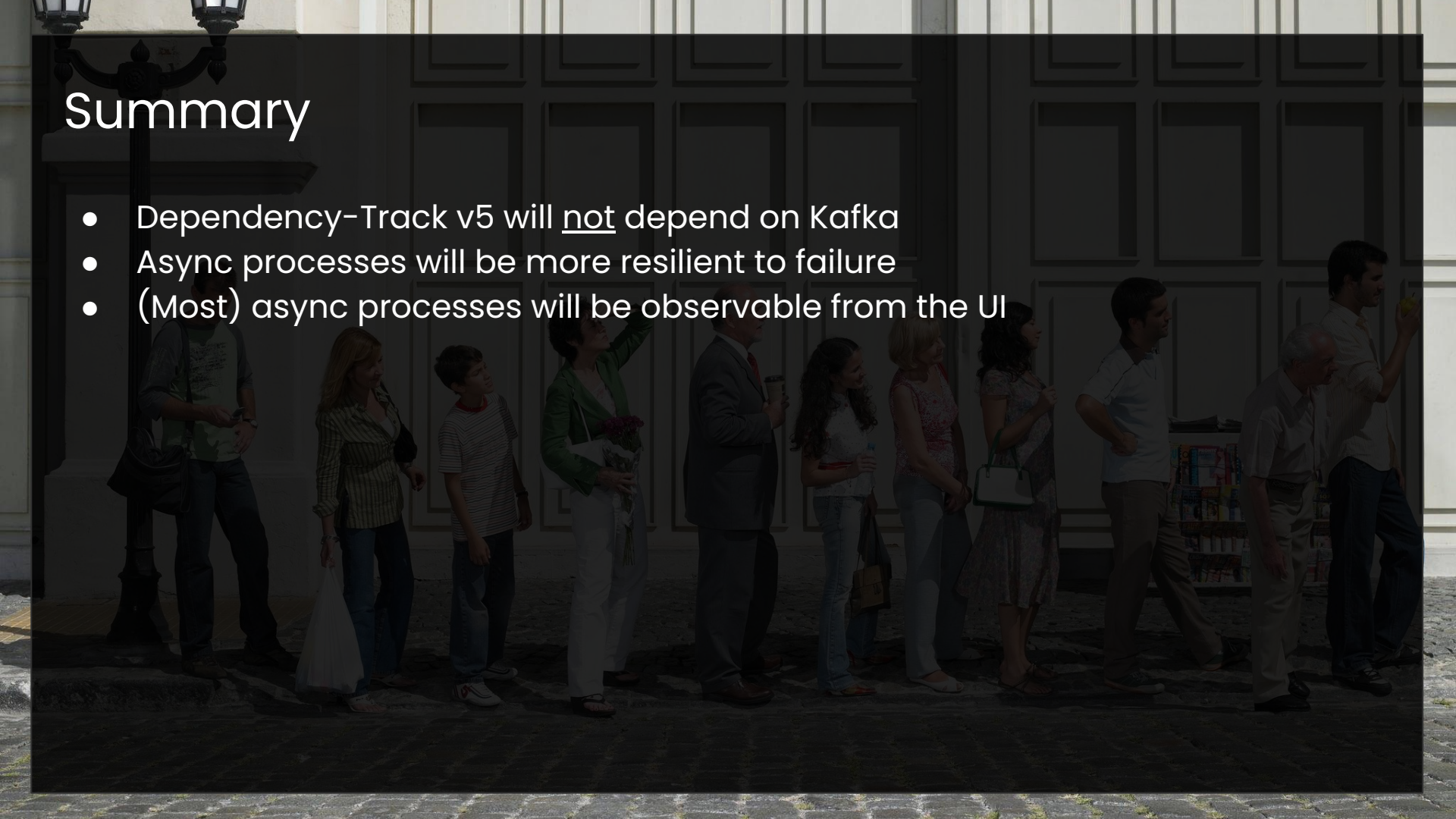
Related: Notification Publishing

ADR-003
Notification Publishing

https://github.com/DependencyTrack/hyades/blob/bc5ac65943f63a6af84eb35d92fc13a62c10bc27/docs/architecture/decisions/003-notification-publishing.md

Summary

- Dependency-Track v5 will not depend on Kafka
- Async processes will be more resilient to failure
- (Most) async processes will be observable from the UI




DATA RETENTION

Data Retention

- Metrics account for the most amount of data over time
- Need a mechanism to prevent it from growing indefinitely
- Deleting individual records is problematic for Postgres
  - Table bloat, I/O spikes, Replication overhead
- Solution: table partitioning
  - One partition per day, e.g. `PROJECTMETRICS_20250507`
  - To enforce retention, simply drop entire partition
  - No table bloat, no I/O spikes, cheap replication
- Retention span is configurable, defaults to 90 days
- Partitions managed by the application
- Will follow same approach for historic workflows

Data Retention

hyades/#1744

Metrics should use partitioned tables

https://github.com/DependencyTrack/hyades/issues/1744


CONTRIBUTORS

For Contributors

- Larger changes require an Architecture Decision Record (ADR)
- Docker / Podman required for tests
  - We’re making heavy use of testcontainers
- The code base still evolves rapidly, prepare for frequent rebasing


WHAT’S NEXT

What’s next

- Complete transition away from Kafka ASAP
  - Most other changes can be done after v5 release
  - The sooner we can stop supporting two release trains the better
- Flesh out access control improvements
  - Might cause breaking changes
  - Migration paths for v4 -> v5