HL7® FHIR® DevDays

JUNE | 2022

Synchronizing Systems Using Whole-System History

Robin Arnold, IBM Watson Health

HL7 FHIR DevDays International 2022 | Hybrid Edition, Cleveland, OH | June 6–9, 2022 | @HL7 | @FirelyTeam | #fhirdevdays | www.devdays.com

ORGANIZED BY firely | HL7 International

Who am I?

- Robin Arnold
- Performance Engineering Lead for IBM FHIR Server at IBM Watson Health
- Lead developer for the IBM FHIR Server JDBC Persistence Layer
- https://github.com/IBM/FHIR

Learning Objectives

- The FHIR R4 Whole-System History Interaction
  - Understand the REST API
  - The Sort Order Limitation for R4
- Using Whole-System History for Synchronizing Systems
  - Addressing the Sort Order Issue
  - Consistency
- Optimizations for Increasing Throughput
- Deployment Considerations for Scalability

FHIR R4 History Interaction

"*The history interaction can be used to set up a subscription from one system to another, so that resources are synchronized between them*"

Note: Network partitions and other failure scenarios can cause a lot of drama. The downstream system may want to detect duplicate requests to avoid creating additional versions.

http://hl7.org/fhir/r4/http.html#3.1.0

http://hl7.org/fhir/r4/http.html#history

FHIR R4 History Interaction

FHIR R4 Specification:

The history interaction retrieves the history of either a particular resource, all resources of a given type, or all resources supported by the system. These three variations of the history interaction are performed by HTTP GET command as shown:

- Instance: GET [base]/[type]/[id]/_history{?[parameters]&_format=[mime-type]}
- Top Level: GET [base]/[type]/_history{?[parameters]&_format=[mime-type]}
- **Whole System: GET [base]/_history{?[parameters]&_format=[mime-type]}**


```
{
  "resourceType": "Bundle",
  "type": "history",
  "link": [
    {
      "relation": "next",
      "url": "https://localhost:9443/fhir-server/api/v4/_history?_sort=none&[server-paging-params]"
    },
    {
      "relation": "self",
      "url": "https://localhost:9443/fhir-server/api/v4/_history?_sort=none"
    }
  ],
  "entry": [
    {
      "id": "1",
      "fullUrl": "https://localhost:9443/fhir-server/api/v4/PractitionerRole/17fb1b7570a-05a46362-
46dc-46ff-9ece-f27b13313bc9",
      "resource": {
        "resourceType": "PractitionerRole",
        "id": ...
        …
      },
      "request": {
        "method": "POST",
        "url": "PractitionerRole"
      },
      "response": {
        "status": "201",
        "location": "https://localhost:9443/fhir-server/api/v4/PractitionerRole/17fb1b7570a-
05a46362-46dc-46ff-9ece-f27b13313bc9/_history/1",
        "etag": "W/\"1\"",
        "lastModified": "2022-03-22T13:00:56.970426Z"
      }
    },
    …
}
```


Response Bundle

History Interaction Response - Ordering

FHIR R4: “The return content is a Bundle with type set to history containing the specified version history, **sorted with oldest versions last**”

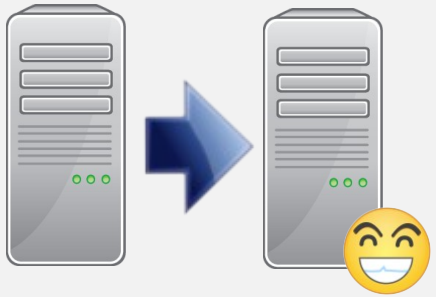
Oldest versions last is reasonable for human consumption

System sync needs to replay changes in order – oldest versions first

Client Choice – Specify Sort Order

```
- _sort=none           => ordering is system-defined
- _sort=_lastUpdated   => ascending by meta.lastUpdated
- _sort=-_lastUpdated => descending by meta.lastUpdated [default]
```

Upstream server must be able to return changes since a known point



History == Change Feed

https://jira.hl7.org/browse/FHIR-31709 (Proposed for FHIR R5)

Implemented in IBM FHIR Server 4.11.0

System Synchronization - Consistency

Making sure downstream systems replicate **all committed changes** but **no uncommitted changes** is non-trivial. In particular, push/notification approaches are sensitive to failures around the transaction boundary. Even distributed transactions aren't completely immune.

Scenario 1:

Update row
Send notification

** APP-SERVER CRASH **

Commit [X]

Scenario 2:

Update row
Commit

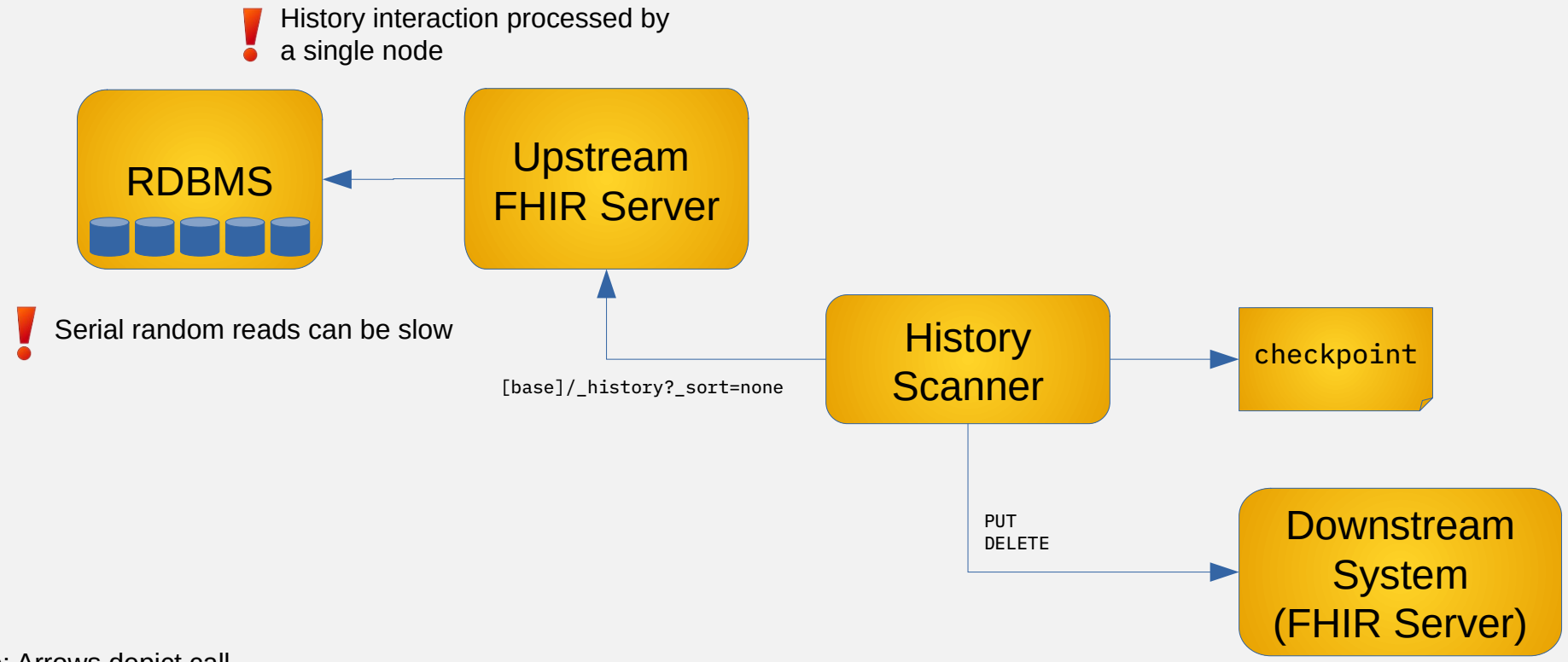
** APP-SERVER CRASH **

Send Notification [X]

If the notification is sent before the transaction is committed, consumers of the event should come back and do a READ (or VREAD) to make sure the data is current.

** This is why pulling changes from an upstream system is often a pragmatic solution

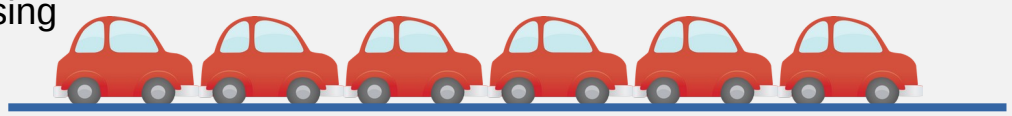
Simple Approach: Single Thread



Note: Arrows depict call not data flow.
For example: A calls B

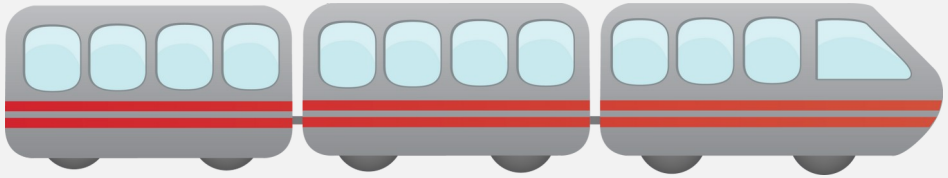
Use Concurrency: Scan Then Fetch

Serial processing of requests is inefficient



Request history without the resource

```
Prefer: return=minimal
[base]/_history?_sort=none
```



Server only returns meta information about each resource

Efficient sequential scan of changes with compact result

Individual resources can then be read and processed in parallel, batched into bundles

VREAD interactions can be distributed across a cluster

Partitioning scheme must take into account certain ordering constraints

Requires more sophisticated checkpoint tracking

With Header 'Prefer: return=minimal'

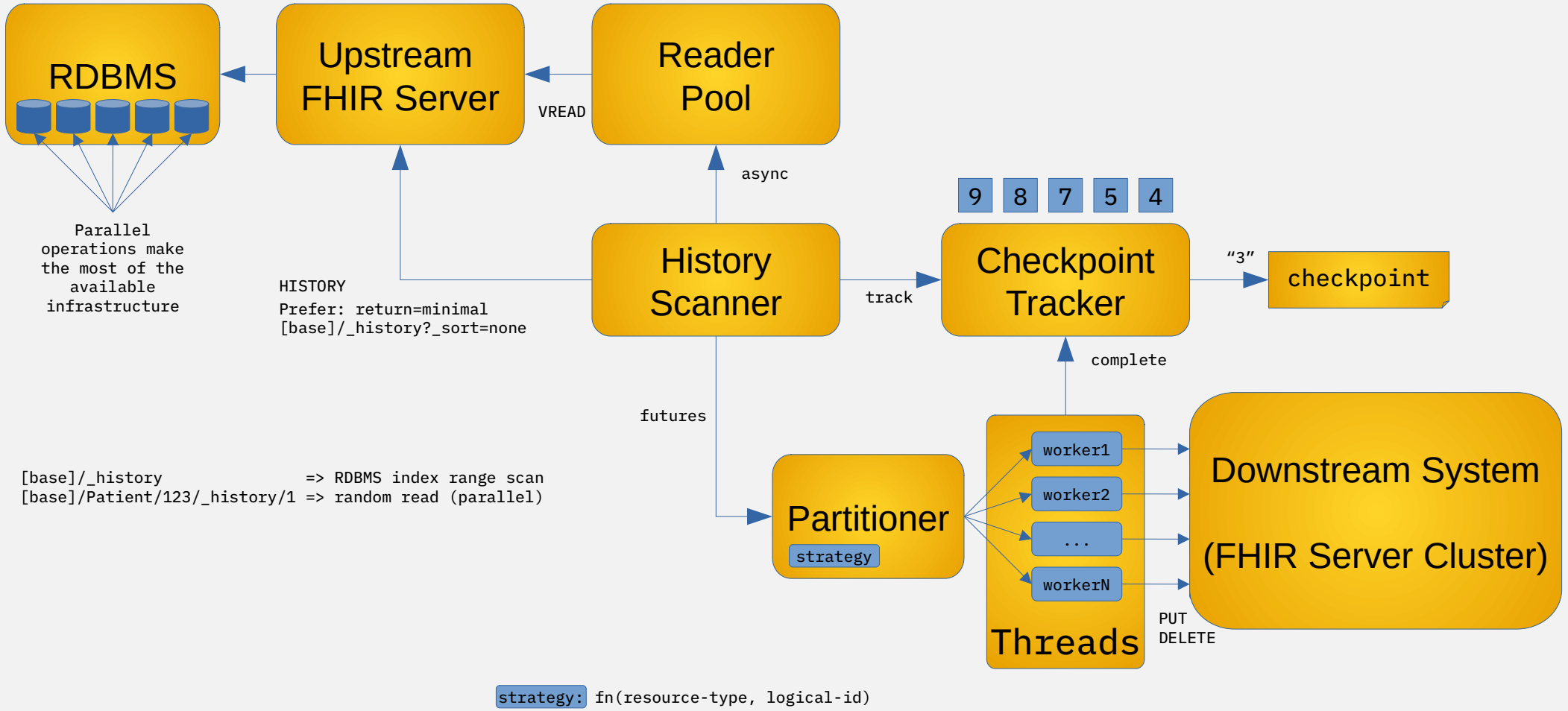
```
curl -X GET \
-H 'content-type: application/fhir+json' \
-H 'Prefer: return=minimal' \
-u 'your-user:your-password' \
-k -i 'https://localhost:9443/fhir-server/api/v4/_history'
```


```
{
  "resourceType": "Bundle",
  "type": "history",
  "link": [
    {
      "relation": "next",
      "url": "https://localhost:9443/fhir-server/api/v4/_history?_sort=none&_changeIdMarker=100"
    },
    {
      "relation": "self",
      "url": "https://localhost:9443/fhir-server/api/v4/_history?_sort=none"
    }
  ],
  "entry": [
    {
      "id": "1",
      "fullUrl": "https://localhost:9443/fhir-server/api/v4/PractitionerRole/17fb1b7570a-05a46362-46dc-46ff-9ece-f27b13313bc9",
      "request": {
        "method": "POST",
        "url": "PractitionerRole"
      },
      "response": {
        "status": "201",
        "location": "https://localhost:9443/fhir-server/api/v4/PractitionerRole/17fb1b7570a-05a46362-46dc-46ff-9ece-f27b13313bc9/_history/1",
        "etag": "W/\"1\"",
        "lastModified": "2022-03-22T13:00:56.970426Z"
      }
    },
    ...
  ]
}
```

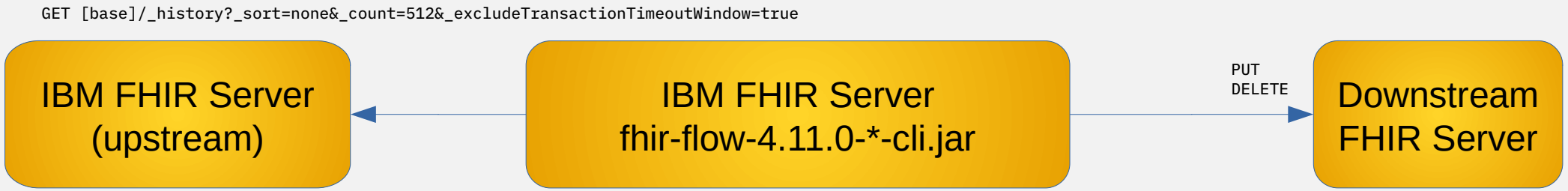

Next links can include server-specific parameters to support pagination: use link to fetch next page of history

fullUrl cannot include version, so we include location in the response

Async VREAD and Parallel Processing

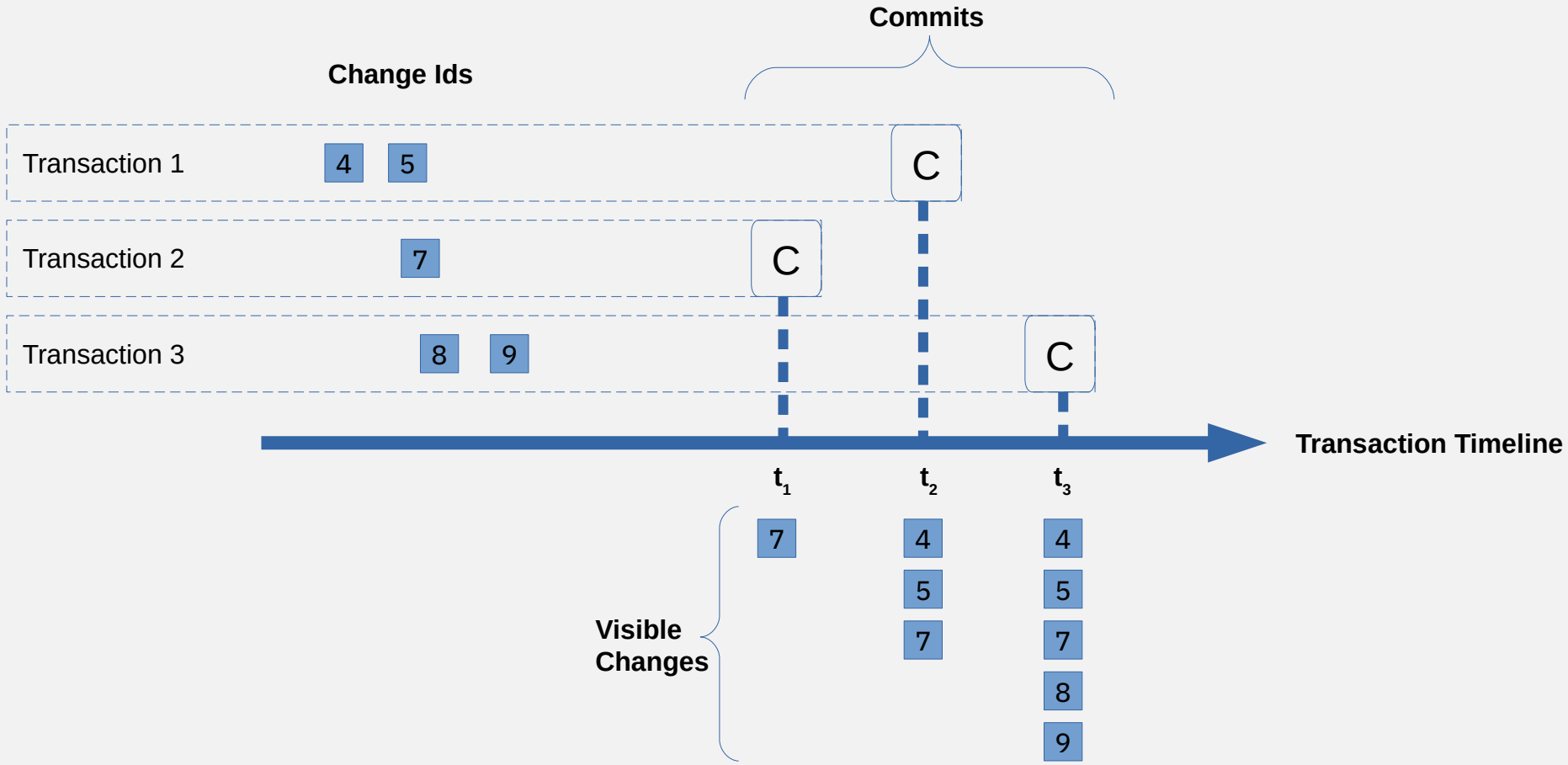


Experimental: fhir-flow




```
FETCH _history?_sort=none&_count=512&_excludeTransactionTimeoutWindow=true
ENQUEUE VREAD Claim/1748a37eaaf-3d45aedd-4675-453e-9fbf-ce9ee7ee4ac4/_history/1
RUNNING VREAD Claim/1748a37eaaf-3d45aedd-4675-453e-9fbf-ce9ee7ee4ac4/_history/1
SUBMIT: CREATE_OR_UPDATE Claim/1748a37eaaf-3d45aedd-4675-453e-9fbf-ce9ee7ee4ac4/_history/1
ENQUEUE VREAD Claim/1748a37eaaf-ffd916f1-f86c-42b4-9c6a-e1092f50ce09/_history/1
SUBMIT: CREATE_OR_UPDATE Claim/1748a37eaaf-ffd916f1-f86c-42b4-9c6a-e1092f50ce09/_history/1
RUNNING VREAD Claim/1748a37eaaf-ffd916f1-f86c-42b4-9c6a-e1092f50ce09/_history/1
ENQUEUE VREAD Condition/1748a37eaaf-8e9461e4-3804-4203-8e11-bde842cf5856/_history/1
…
FETCH _history?_sort=none&_count=512&_changeIdMarker=1023&_excludeTransactionTimeoutWindow=true
…
FETCH _history?_sort=none&_count=512&_changeIdMarker=1967&_excludeTransactionTimeoutWindow=true
```

Handling Continuous Data Flows



Solution: Do not fetch any changes which fall inside the current transaction timeout window

Guarantees changes will always appear in order as required for a given {resource-type, logical-id}

Similarly, be aware of clock drift in clusters when using lastUpdated

(*) obviously depends on the underlying database implementation

What if there are dependencies between resources?

Dependent resources may be routed to another partition;

Possible race condition if a resource is PUT before the resources it depends on

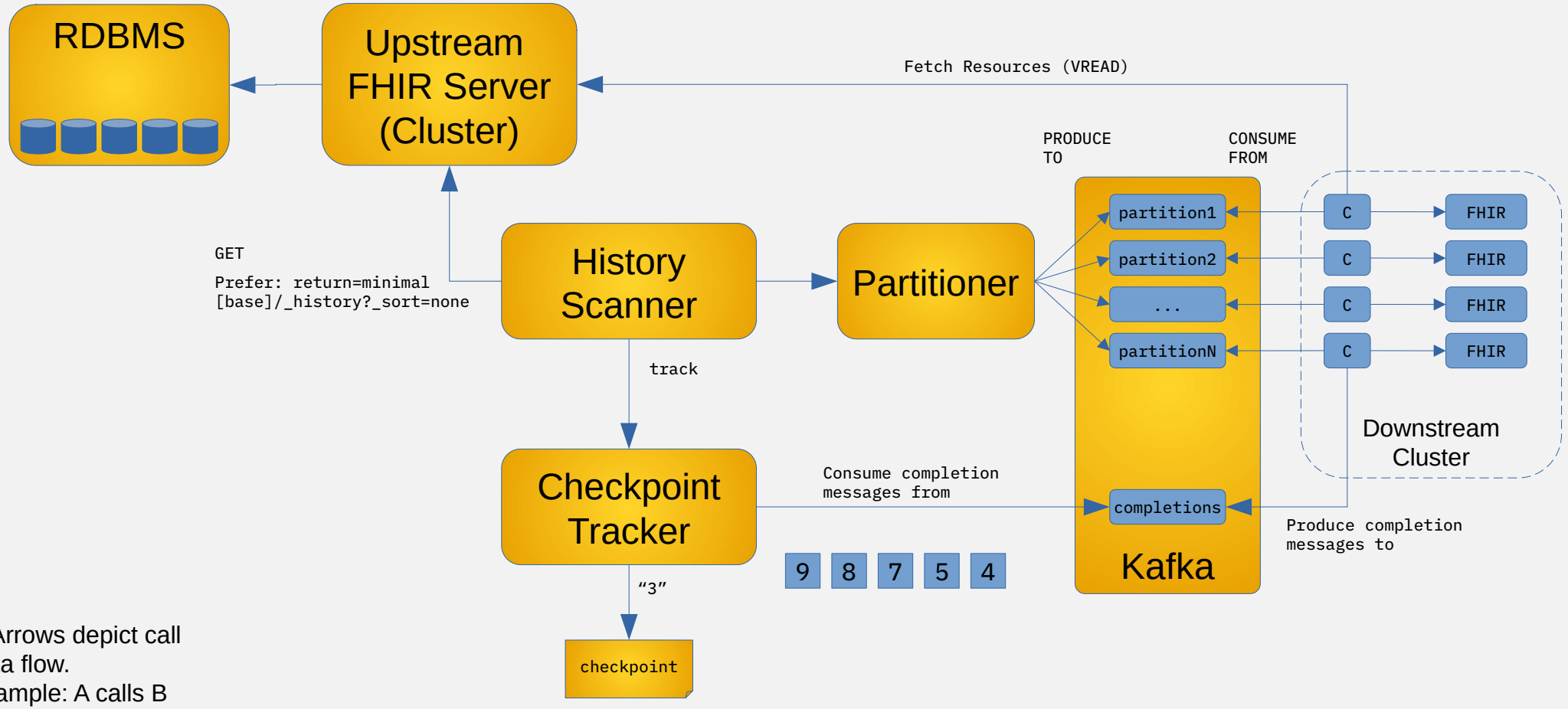
---

Catch the error and retry

Will eventually succeed

No chance of deadlock because ordering is consistent within a partition (oldest to newest)

Scaling Out



Note: Arrows depict call not data flow.
For example: A calls B

Resources

fhir-flow	https://github.com/IBM/FHIR/tree/main/fhir-flow
Schema Design	https://github.com/IBM/FHIR/blob/main/fhir-persistence-schema/docs/SchemaDesign.md
Example sending parameters to Kafka (in feature development)	https://github.com/IBM/FHIR/blob/robin-citus-1/fhir-server/src/main/java/com/ibm/fhir/server/index/kafka/FHIRRemoteIndexKafkaService.java
FHIR History	http://hl7.org/fhir/r4/http.html#history

What did you learn?

- Current capabilities of the FHIR Whole System History Interaction
- How to use the history endpoint as a source for system synchronization
- API changes required to support a practical implementation
- Optimizations to improve throughput and scalability

Contact

- During DevDays, you can find / reach me here:
  - Via Whova App – Speaker's Gallery
  - Email: robin.arnold@ibm.com
  - LinkedIn: https://www.linkedin.com/in/robin-arnold-50a383
  - Chat: https://chat.fhir.org/#narrow/stream/212434-ibm
  - https://github.com/IBM/FHIR

Q&A

ORGANIZED BY