APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 26-28 SEPTEMBER 2016

Can we run the whole Web on Apache Sling?

Bertrand Delacretaz & Chetan Mehrotra
@bdelacretaz - @chetanmeh
Sling committers and PMC members
CQ/AEM core team members, Adobe

slides revision 2016-09-26

Are you guys crazy?

(yes, but not that much)

The Whole Web ??

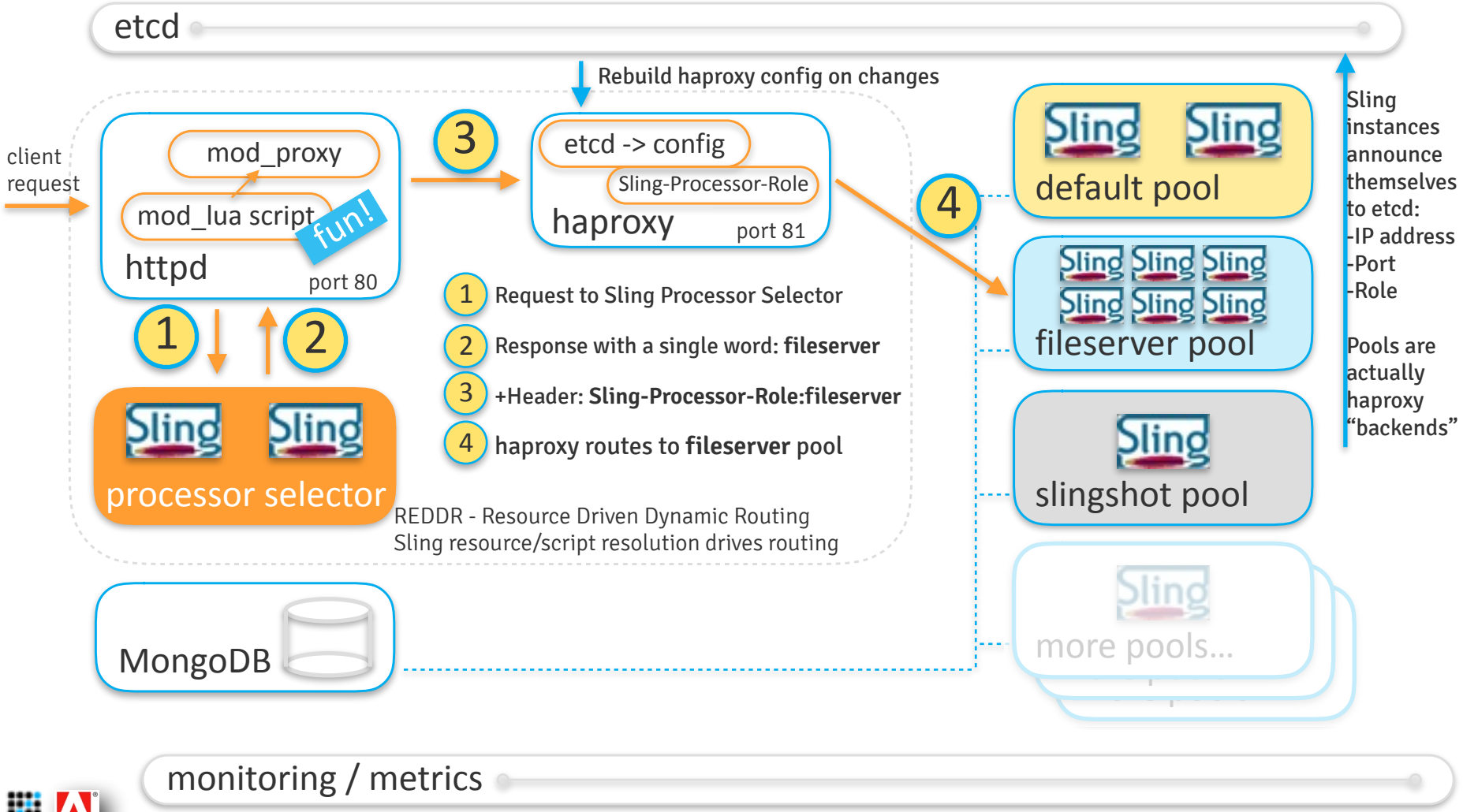
Where's Sling going?

from

SERVERS

to

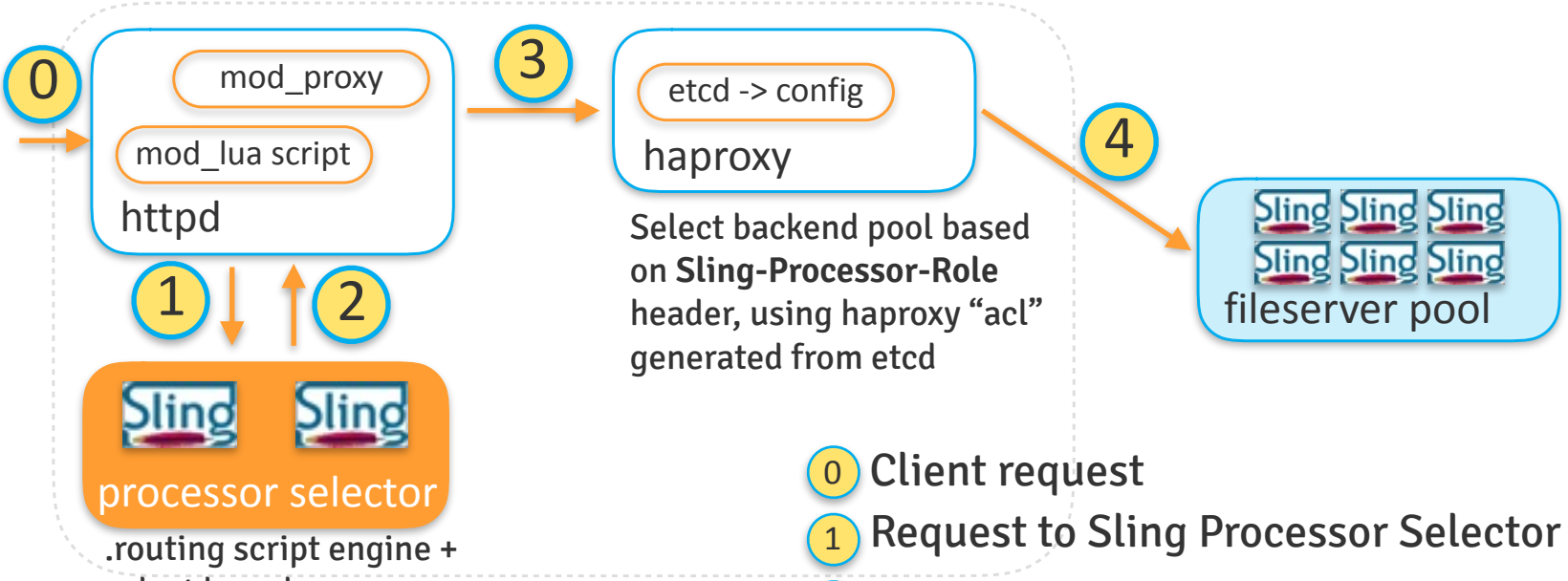
SYSTEMS


etcd
Rebuild haproxy config on changes
client request
mod_proxy
mod_lua script
fun!
httpd
port 80
3
etcd -> config
Sling-Processor-Role
haproxy
port 81
4
default pool
fileserver pool
slingshot pool
more pools...
Sling instances announce themselves to etcd:
-IP address
-Port
-Role
Pools are actually haproxy "backends"
1
2
processor selector
1 Request to Sling Processor Selector
2 Response with a single word: fileserver
3 +Header: Sling-Processor-Role:fileserver
4 haproxy routes to fileserver pool
REDDR - Resource Driven Dynamic Routing
Sling resource/script resolution drives routing
MongoDB
monitoring / metrics

REDDR

REsource-Driven Dynamic Routing

REDDR routing to fileserver instances pool



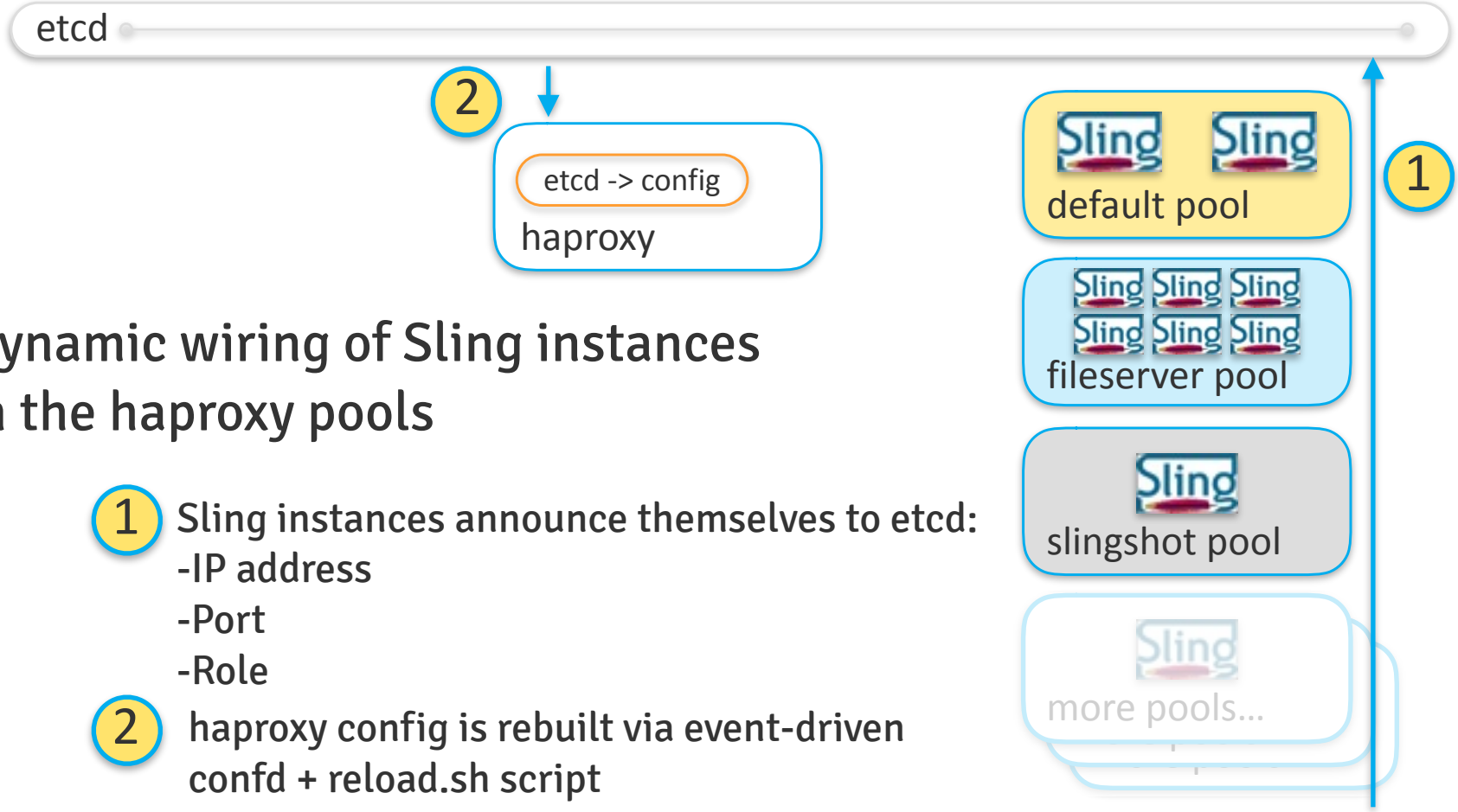
Select backend pool based on **Sling-Processor-Role** header, using haproxy "acl" generated from etcd

.routing script engine + select based on resource type and parent types. Default servlets and scripts disabled.

0. Client request
1. Request to Sling Processor Selector
2. Response with a single word: **fileserver**
3. +Header: **Sling-Processor-Role:fileserver**
4. haproxy routes to **fileserver** pool

Dynamic Registration of Sling Instances

regenerate haproxy config based on etc data



Dynamic wiring of Sling instances in the haproxy pools

1. Sling instances announce themselves to etcd:
   - -IP address
   - -Port
   - -Role
2. haproxy config is rebuilt via event-driven confd + reload.sh script

Building customised Sling Docker images

with just a provisioning model + trivial Dockerfile

Base and default-processor images

```
base/pom.xml
base/src/main/docker/Dockerfile
base/src/main/docker/slingroot/announce.sh
base/src/main/docker/slingroot/start.sh
base/src/main/docker/slingroot/wait-for-it.sh

default-processor/pom.xml
default-processor/src/main/docker/Dockerfile
default-processor/src/main/provisioning/composum-browser.txt
default-processor/src/main/provisioning/default-processor.txt
default-processor/src/main/provisioning/launchpad.txt
default-processor/src/main/provisioning/…
```

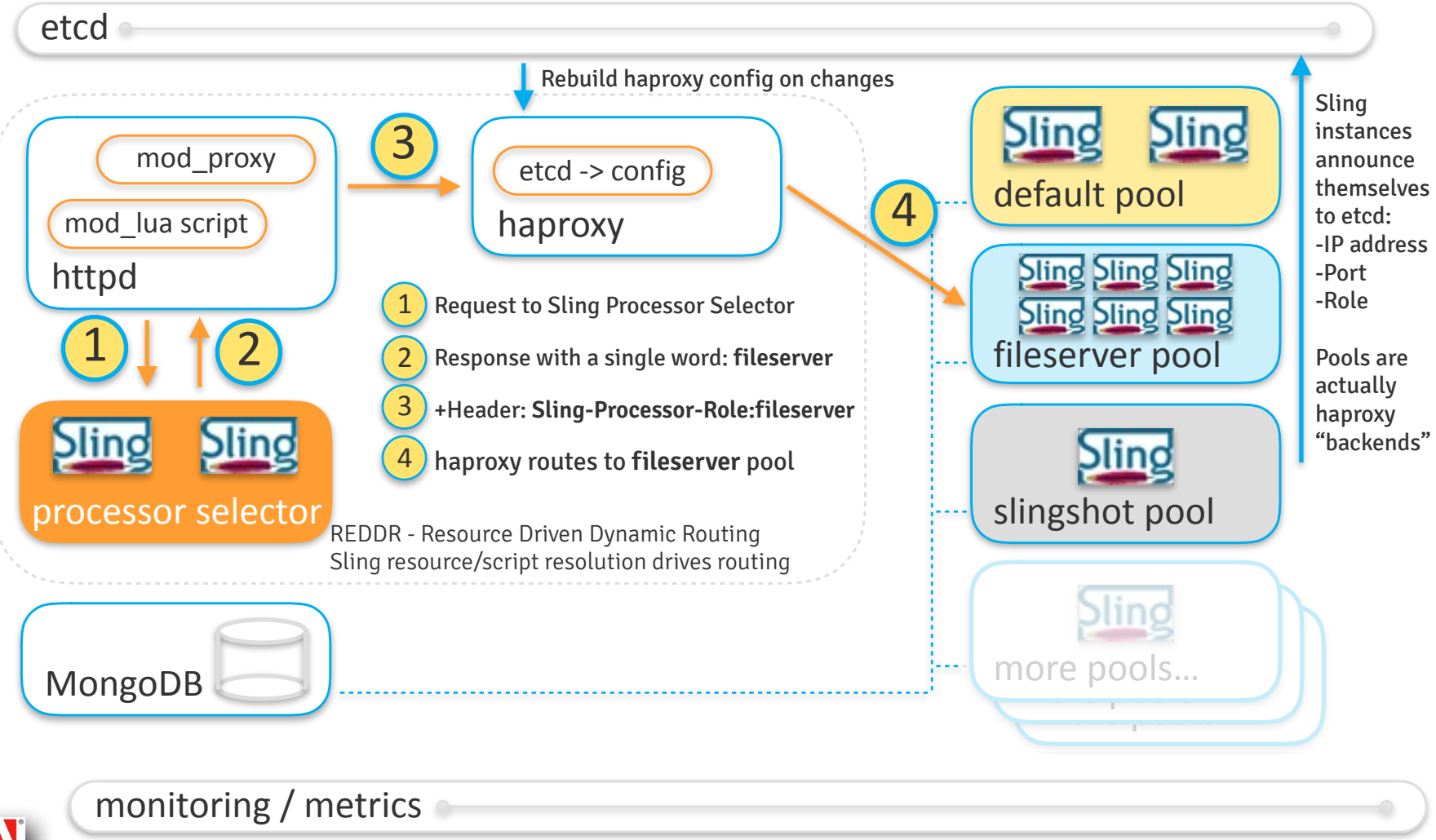
Custom Sling Docker images

Dockerfile for default-processor

```
FROM ch.x42.at16.base
COPY ${project.build.finalName}.jar /opt/sling/launchpad.jar
```

DYNAMIC CLUSTER DEMO

A Big Sling Cluster…on my laptop

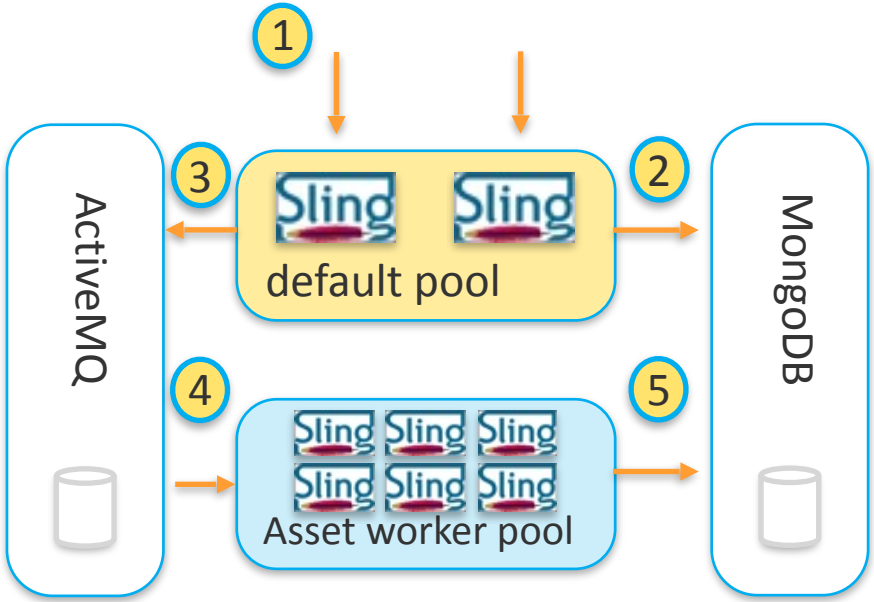

etcd
Rebuild haproxy config on changes
mod_proxy
mod_lua script
httpd
3
etcd -> config
haproxy
4
default pool
fileserver pool
slingshot pool
more pools...
Sling instances announce themselves to etcd:
-IP address
-Port
-Role
Pools are actually haproxy "backends"
1
2
processor selector
1 Request to Sling Processor Selector
2 Response with a single word: fileserver
3 +Header: Sling-Processor-Role:fileserver
4 haproxy routes to fileserver pool
REDDR - Resource Driven Dynamic Routing
Sling resource/script resolution drives routing
MongoDB
monitoring / metrics

“Background Worker” Sling instance

minimal, disposable, focused

Worker Profile

- Tailor made Sling instance
- Min required bundles
- Low footprint
- JCR API used mostly for CRUD
- No observation
- Disposable
- Take post processing load off frontend servers



1. Request to Sling Frontend server
2. Write to NodeStore
3. Add job to message queue
4. Pick up of job by one of the worker
5. Write back of Job result

Repository Etiquette

be precise while talking to repository and thou shall not be troubled!

Choose the right nodetype

Avoid `nt:unstructured`

- Orderable children
- Limits writes

```
+ dashboard (nt:unstructured)
  - role_observer - projects-outdoors
  - role_editor - projects-editor
  + gadgets (nt:unstructured)
   + team
   + asset
```

Prefer `oak:Unstructured`

```
+ dashboard (oak:Unstructured)
  - role_observer - projects-outdoors
  - role_editor - projects-editor
  + gadgets (oak:Unstructured)
   + team
   + asset
```

OAK-657

Choose the right nodetype

Avoid `nt:resource` with `nt:file`

- Referenceable
- 1 nt:resource = 1 entry in uuid index
- 1M Files = 1M entry in uuid index

```
+ book.jpg (nt:file)
 - jcr:createdBy - admin
 + jcr:content (nt:resource)
   - jcr:lastModifiedBy - admin
   - jcr:mimeType - image/jpeg
   - jcr:uuid - "dafe0c9c-1872-4397"
```

Prefer `oak:Resource`

```
+ book.jpg (nt:file)
 - jcr:createdBy - admin
 + jcr:content (oak:Resource)
   - jcr:lastModifiedBy - admin
   - jcr:mimeType - image/jpeg
```

OAK-4567

Query Precisely

- Use specific nodetype
- Relativize property names wrt root of micro tree
- Include path restrictions
- Use union if nodetypes differ

```
SELECT *
FROM [nt:base] AS a
    WHERE
    a.[type] = 'image'
```

```
SELECT *
FROM [dam:Asset] AS a
    WHERE ISDESCENDANTNODE([/content/dam])
    AND a.[jcr:content/metadata/type] = 'image'
```

Observe Precisely

- Observation costs resources
- Use JackrabbitEventFilter to filter on
  - Multiple paths
  - Nodetypes
- In works* filtering based on
  - Property names
  - Node names

```
JackrabbitEventFilter eventFilter
= new JackrabbitEventFilter()
.setAbsPath(paths[0])
.setNodeTypes(new String[]{"dam:Asset"})
.setEventTypes(Event.NODE_ADDED)
.setIsDeep(true);

eventFilter.setAdditionalPaths("/content/en");
JackrabbitObservationManager om =
(JackrabbitObservationManager)session.getWorkspace()
.getObservationManager();
om.addEventListener(this, eventFilter);
```

*OAK-4796

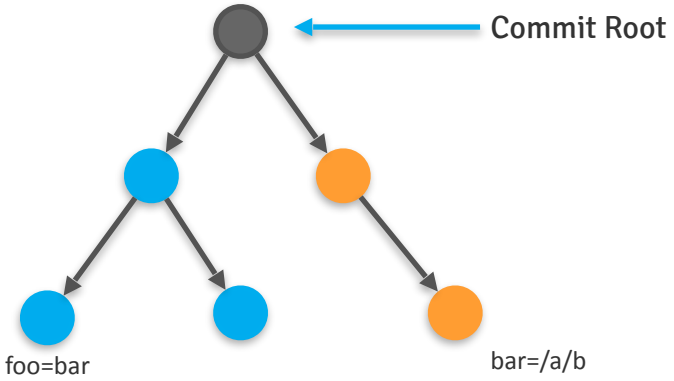
Define your own types

- Nodetype/mixin are content annotation
- Repository uses them to
  - Enforce structure
  - Determine index rules
  - Filter observation events
  - Bundle Nodes* (new stuff!)

*OAK-1312

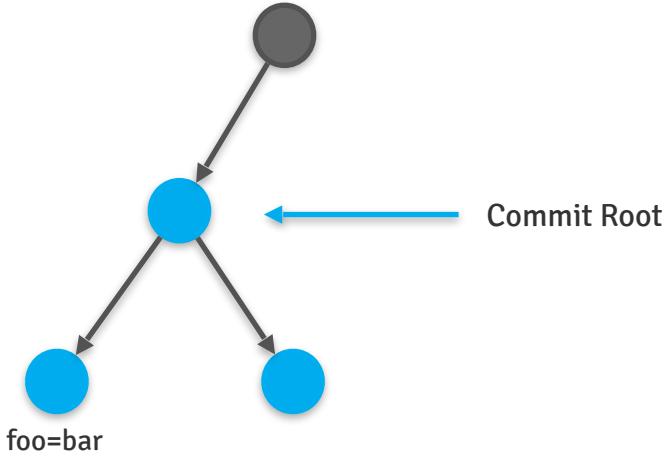
Prefer Lucene Property Indexes

- Property Indexes
  - Cause conflict in index data
  - Causes commit root to be ‘/’
  - Stored as nodes
  - Good for sparse and full sync case



Prefer Lucene Property Indexes …

- Lucene Property Indexes
  - Async indexing
  - No impact on commit root
  - Compact storage
  - Multi restriction evaluation
- Are not they async?
  - In works near real time indexing*



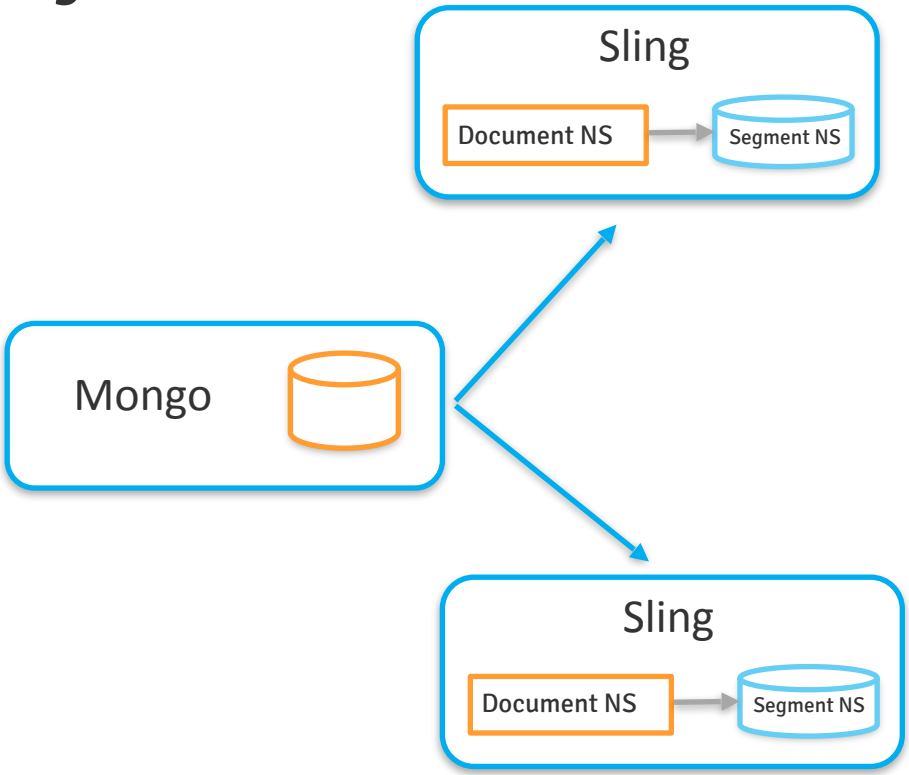
*OAK-4412

FUTURE PERFORMANCE-RELATED Oak FEATURES

work in progress!

Segment NS as Local Secondary Store

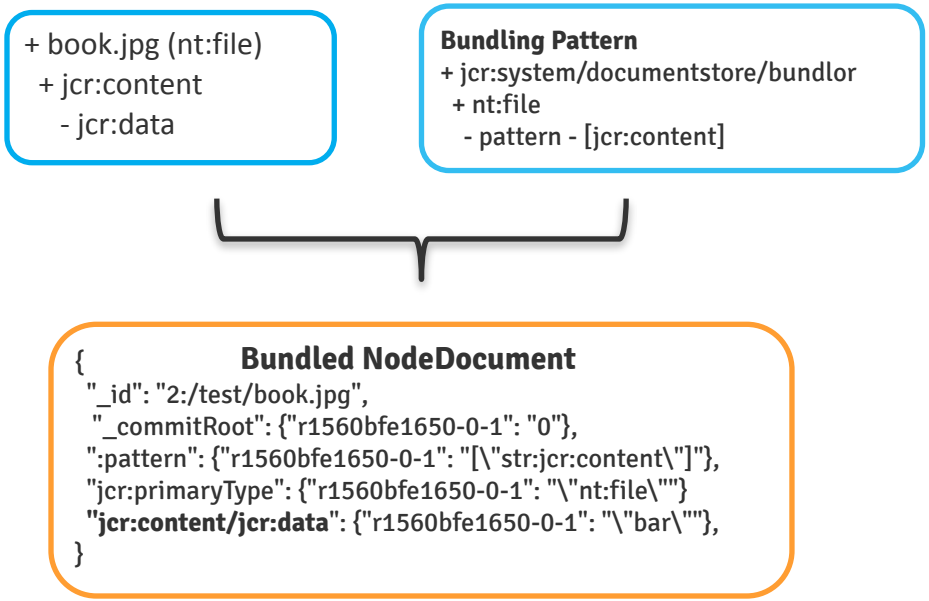
- Segment NS act a local copy of remote repository like a local git repo
- Updated via Observation
- Handles read call for “not so recently modified” Nodes
- Reduces read latency
- Configured to store certain path. Defaults to ‘/’



*OAK-4180

Bundle Multiple JCR Node in one Document

- Store subtree aggregates of specific nodes in same NodeDocument
- JCR Node to Mongo mapping
  - 1 JCR Node = 1 Mongo Doc
  - 1 dam:Asset ~ 20 JCR Node
  - 1M Assets = 20M Mongo Doc
- Nodetype as content annotation hint to optimize storage
- Benefits
  - Lower size of _id index
  - Less number of queries to read the micro tree



*OAK-1312

CODA

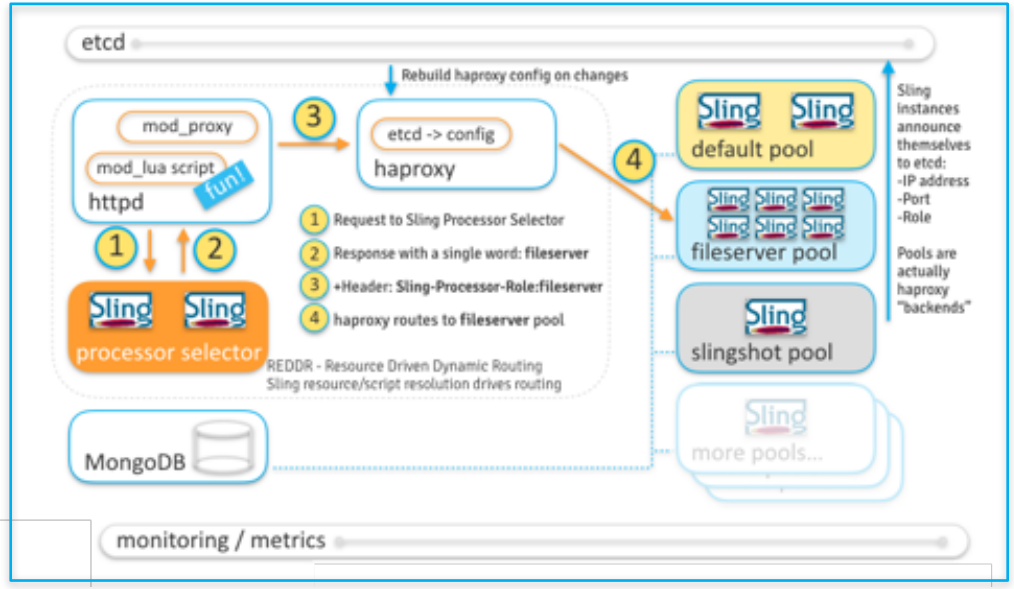
The Whole Web, really?

CODA

REDDR enables (extreme) scaling driven by Sling resource/script resolution.

Building customized Sling instances is easy: provisioning model + trivial Dockerfile.

From servers to systems!



Precise and focused repository types and operations improve performance.

Thank you for attending!
Chetan Mehrotra (@chetanmeh)
Bertrand Delacretaz (@bdelacretaz)