Buzzword-free RESTful container-based OSGi middleware for NoSQL databases

I hope you still have some air in your lungs after reading this title.

Bertrand Delacrétaz
@bdelacretaz, grep.codeconsult.ch

Principal Scientist, Adobe Basel
Apache Software Foundation Member and Director

Thanks to Robert Munteanu, Adobe, for the collaboration on the demo.

Snowcamp Grenoble, January 2016

slides revision 2016-01-21

Adobe

warning

this talk is about

INTEGRATION

so no details on Docker, Sling or the NoSQL databases that we use.

But it’s fun integrating all those things.

so

what

are we building?

Apache Sling HTTP front-end

Mongo ResourceProvider

Redis ResourceProvider

Planets ResourceProvider

OSGi configs

OSGi framework

Docker container

HTTP / JSON

Redis server

Docker container

MongoDB server(s)

Docker container

uniclient

Docker container

Docker events

dbwatcher

Docker container

Activation of ResourceProvider for new DB containers

Command line for testing

(It's at https://github.com/bdelacretaz/sling-nosql-frontend)

Apache Sling in Docker

```
# Dockerfile
FROM java:8
COPY sling.jar /opt/sling/sling.jar
WORKDIR /opt/sling/
EXPOSE 8080
ENV JAVA_OPTS -Xmx512m
CMD exec java $JAVA_OPTS -jar sling.jar
```

```
# docker-compose.yml snippet
sling:
    image: sling-nosql-demo:latest
     container_name: sling
     ports:
          - "8080:8080"
```

Mongo & Redis in docker-compose.yml

```
mongo:
    image: mongo:3.0
    container_name: mongo
    ports:
        - "27017:27017"

redis:
    image: redis:3.0.6
    container_name: redis
    ports:
        - "6379:6379"
```

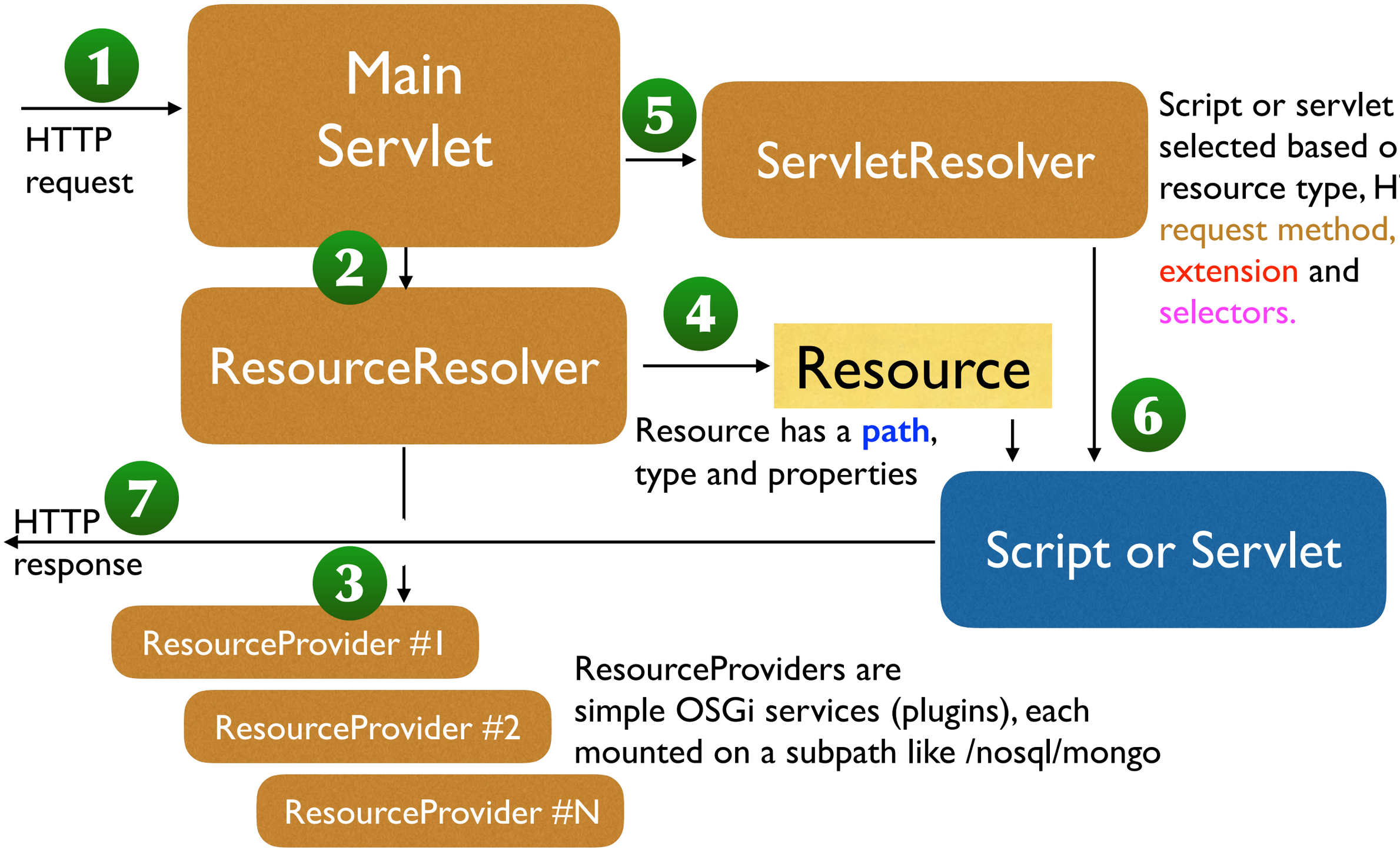
demo

Gentlemen, start your engines!

aka demo1
“the planets”

The Sling HTTP processing pipeline

GET http://example.com/products/bottle.print.a4.html



ResourceProviders are simple OSGi services (plugins), each mounted on a subpath like /nosql/mongo

Sling ResourceProvider service API

Slightly simplified here

```
/** Register such an OSGi service with
 *  a “provider.roots” property to
 *  mount it in the Sling resource tree
 */
interface ResourceProvider {

 Resource getResource(
   ResourceResolver resourceResolver,
   String path);

  Iterator<Resource> listChildren(
    Resource parent);
}
```

PlanetsResourceProvider service

Creating an OSGi config creates such a service

```
@Component(
        metatype=true,
        configurationFactory = true,
        policy = ConfigurationPolicy.REQUIRE
)
@Service
@Properties({
        @Property(name=ResourceProvider.ROOTS)
})
public class PlanetsResourceProvider
  implements ResourceProvider {
…
```

OSGi configs drive services

and

ResourceProvider provides content

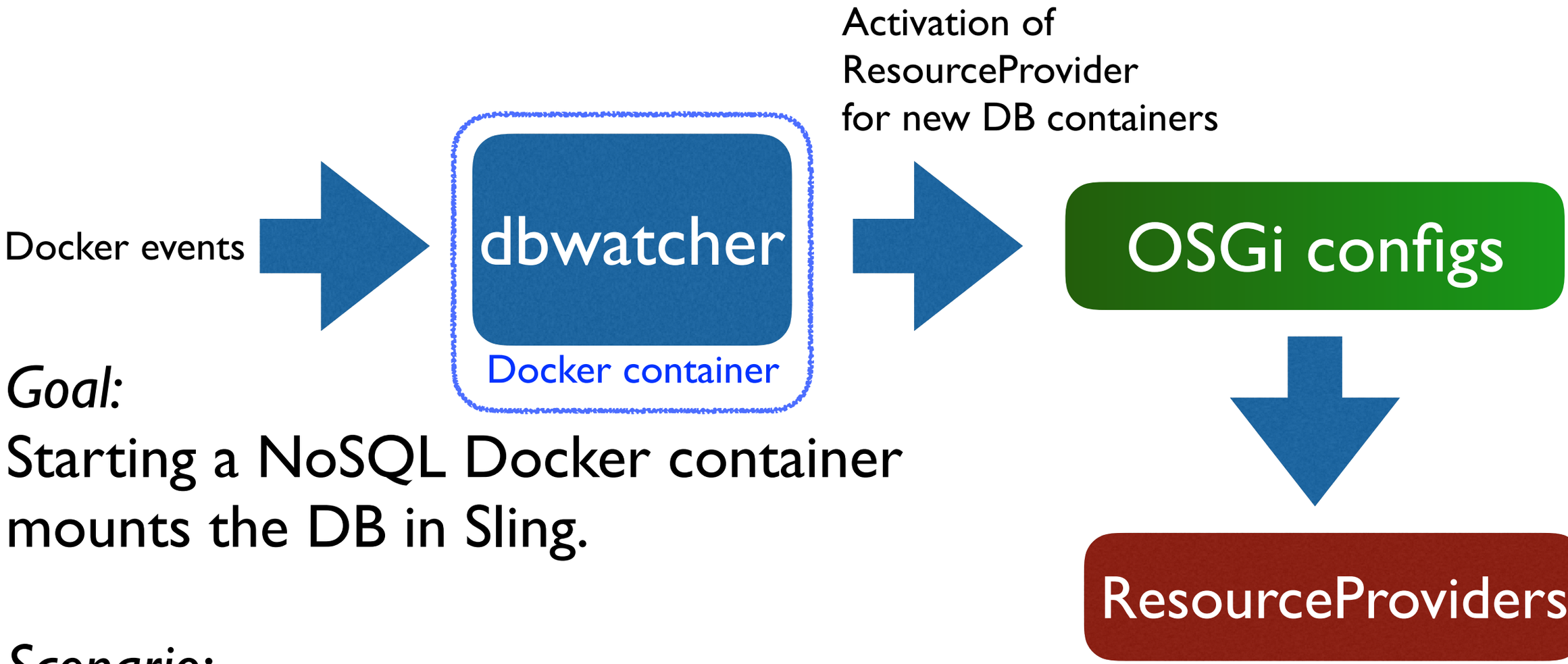
that’s what we have so far

demo

Didn’t we say “noSQL”?

aka demo2
“Mongo and Redis with static configs”

How about dynamic DB mounts?



Goal:
Starting a NoSQL Docker container mounts the DB in Sling.

Scenario:
Get Docker "container start" event.
Does the container have our "DB type label"?
If yes, get more info: DB type, IP address, etc
And inject an OSGi config in Sling, to instantiate a suitable ResourceProvider

demo

Mounty Dynamic is in town

aka demo3
“Dynamic mounts of new NoSQL containers”

Docker events at the command line

```
2016-01-20T14:46:53.604770204Z fe31d475:
(from busybox) create

2016-01-20T14:46:53.679080597Z fe31d475:
(from busybox) attach

2016-01-20T14:46:53.707154497Z fe31d475:
(from busybox) start

2016-01-20T14:46:53.859539615Z fe31d475:
(from busybox) resize

2016-01-20T14:46:53.970689118Z fe31d475:
(from busybox) die
```

docker-watcher script, simplified

```
# Extract container ID from docker start event
# And use docker inspect to retrieve labels, IP etc. from container

docker events -f event=start | sed -u 's/[^ ]*//' | while read ID
do
 docker inspect $ID > $TMPFILE
 DBTYPE=$(jq '.[0].Config.Labels["ch.x42.slingdb.type"]' < $TMPFILE)
 HOST=$(jq .[0].NetworkSettings.Networks.docker.IPAddress < $TMPFILE)
 PORT=$(jq '.[0].Config.Labels["ch.x42.slingdb.port"]' < $TMPFILE)
 NAME=$(jq .[0].Name < $TMPFILE | sed 's/[^a-zA-Z0-9_]//g')

 # create_mongo_config.sh for example injects an OSGi config
 # in Sling for the Mongo ResourceProvider, via HTTP
 CMD="create_${DBTYPE}_config.sh $HOST $PORT /nosql/$NAME"
 echo "Container $ID started, running $CMD"
 ${MYFOLDER}/${CMD} < /dev/null
done
```

IF U CN RD THS YU HV GD EYS

Coda

Combining the dynamics of containers with configuration-driven OSGi ResourceProvider services allows us to create a dynamic system.

Apache Sling provides a nice way to aggregate data from very different data sources.

Sling's rendering features can be used to reformat or present the results.

Code:
https://github.com/bdelacretaz/sling-nosql-frontend
See also:
http://sling.apache.org
I'm @bdelacretaz - thanks!